VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2021 Bc. Miroslav Zelený

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

AUTOMATIZOVANÉ SLEDOVÁNÍ POHYBUJÍCÍCH SE OBJEKTŮ POMOCÍ ROBOTICKÉHO MANIPULÁTORU
AUTOMATED OBJECT TRACKING USING ROBOTIC MANIPULATOR

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Miroslav Zelený**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Adam Chromý, Ph.D.**
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Kybernetika, automatizace a měření**

Ústav automatizace a měřicí techniky

Student: Bc. Miroslav Zelený ***ID:*** 192301
Ročník: 2 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Automatizované sledování pohybujících se objektů pomocí robotického manipulátoru

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je vytvořit systém, který bude umět pomocí robotického manipulátoru otáčet kamerou tak, aby vybraný objekt stále zůstával v centru záběru.

1. Seznamte se s robotickým manipulátorem Epson C3, jeho programováním a stávajícími systémy pro online ovládání. Seznamte se s předloženou kamerou a způsobem získávání naměřených dat.
2. Navrhněte algoritmus, který v obraze vyhledá významné body, určí jejich polohu v prostoru a na základě změny této polohy mezi dvěma snímky vypočte požadovanou změnu polohy robotického manipulátoru tak, aby sledovaný objekt zůstal v centru záběru.
3. Pro medicínské využití zpřesněte algoritmus z bodu 2 využitím lokalizačních markerů. Navrhněte, jak mají vypadat a ověřte jejich výsledný vliv na přesnost udržování konstantního záběru kamery.
4. Využitím algoritmu z bodů 2 a 3 vytvořte funkční třídu pro sledování objektů, jejímž vstupem budou data ze snímačů a aktuální poloha manipulátoru a výstupem nová poloha manipulátoru. Funkčnost třídy dokažte demonstrační aplikací.
5. Realizujte experimenty, které prokáží využitelnost systému v medicínském skenování nebo profesionálním fotografování.

DOPORUČENÁ LITERATURA:

S. Challa, M. R. Morelande, D. Mušicki, a R. J. Evans, Fundamentals of Object Tracking. Cambridge University Press, 2011.

Termín zadání: 8.2.2021 ***Termín odevzdání:*** 17.5.2021

Vedoucí práce: Ing. Adam Chromý, Ph.D.

doc. Ing. Petr Fiedler, Ph.D.
předseda rady studijního programu

Abstrakt

Tato diplomová práce se zabývá sledováním objektů za pomoci robotického manipulátoru Epson C3 a barevné kamery. Práce popisuje základní vlastnosti zařízení, které mají být použity. Ze softwarových nástrojů pro počítačové vidění je použita knihovna OpenCV, respektive její wrapper EmguCV. Je zde probrána základní problematika a principy sledování objektů v obraze a jsou zde představeny některé metody sledování. Tyto metody byly testovány a jsou zde tedy uvedeny jejich silné i slabé stránky, které se během testování projevily. Dále je zde uveden postup pro výpočet nových souřadnic kamery a efektoru manipulátoru pomocí homogenních transformací. Práce obsahuje výsledky testování daných algoritmů a jejich vyhodnocení. Výstupem práce je testovací aplikace pro robot Epson C3.

Klíčová slova

OpenCV, EmguCV, Epson C3, homogenní transformace, sledování objektů, sledovací algoritmy, porovnávání významných bodů, markery, C#, detekce v obraze, zpracování obrazu, robotický manipulátor

Abstract

This diploma thesis deals with the tracking of objects using a robotic manipulator Epson C3 and a color camera. The work describes the basic qualities of the device to be used. The OpenCV library and its wrapper EmguCV are used as software tools for computer vision. It discusses the basic issues and principles of tracking objects in the image and introduces some methods of tracking. These methods have been tested and therefore their strengths and weaknesses, which appeared during testing, are listed here. Furthermore, there is a procedure for calculating the new coordinates of the camera and the manipulator effector using homogeneous transformations. The work contains the results of testing the algorithms and their evaluation. The output of the work is a test application for the Epson C3 robot.

Keywords

OpenCV, EmguCV, Epson C3, homogeneous transformations, object tracking, tracking algorithms, comparison of significant points, markers, C #, image detection, image processing, robotic manipulator

Bibliografická citace

ZELENÝ, Miroslav. *Automatizované sledování pohybujících se objektů pomocí robotického manipulátoru* [online]. Brno, 2021 [cit. 2021-05-17]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/134588. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce Adam Chromý.

Prohlášení autora o původnosti díla

Jméno a příjmení studenta:	*Bc. Miroslav Zelený*
VUT ID studenta:	*192301*
Typ práce:	*Diplomová práce*
Akademický rok:	*2020/21*
Téma závěrečné práce:	*AUTOMATIZOVANÉ SLEDOVÁNÍ POHYBUJÍCÍCH SE OBJEKTŮ POMOCÍ ROBOTICKÉHO MANIPULÁTORU*

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

.

V Brně dne: 17. května 2021 ----------

podpis autora

Poděkování

Děkuji vedoucímu diplomové práce Ing. Adamovi Chromému, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc¸ za cenné rady, trpělivost a čas, který mi při zpracování mé diplomové práce věnoval.

V Brně dne: 17. května 2021 ----------

podpis autora

Obsah

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

Úvod

Fenomén robotických manipulátorů se v dnešní době netýká jen strojírenských oborů. V dnešní době se začínají uplatňovat i v jiných oborech než-li jsou montážní linky automobilových závodů, obráběcí linky nebo jiné automatizované činnosti v rámci výrobních závodů.

Jako příklad jednoho z nekonvenčních využití robotického manipulátoru může sloužit i projekt RoScan. V rámci tohoto projektu se za pomoci robotického manipulátoru provádí skenování povrchu a následné mapování dalších informací na tento model, jako jsou informace o barvě a teplotě. Projekt nachází uplatnění hlavně jako diagnostický nástroj na poli medicíny, kde pomáhá diagnostikovat onemocnění, která se projevují změnou teploty na snímané části těla.

Hlavním cílem této práce je vytvořit algoritmus, který by udržel snímaný objekt ve středu plochy snímané pomocí kamery umístěné na manipulátoru, popřípadě u toho udržoval konstantní vzdálenost od snímaného objektu. Algoritmus by měl být směřován tak, aby ho bylo popřípadě možné přidat do projektu RoScan. Požadavek na udržení sledovaného objektu jako vycentrovaného má reálné opodstatnění a je velmi důležitý, pokud chceme snímat teplotu po delší časový úsek, kdy není možné, aby zůstal pacient v naprostém klidu. V některých případech to není ani fyzicky možné, například když pacient musí pumpovat s rukou, aby bylo možné sledovat prokrvení končetiny.

Celý princip sledování spočívá v natáčení kamery pomocí manipulátoru, zatímco budeme pomocí kamery zachycovat určitý objekt zájmu, a tak se snažit přiblížit práci v reálném čase.

Výsledné řešení by zároveň mohlo najít využití i v jiných oblastech než pouze v rámci projektu RoScan. Bylo by ho možné využít i v rámci fotografování, kdy by díky němu mohly vznikat za pomoci robotického manipulátoru fotografie pohybujících se objektů s efektem rozmazaného pozadí, popřípadě by mohl sloužit jednoduše jako mechanická stabilizace obrazu, ať už při focení nebo při pořizování videa. Jako nevýhoda tohoto řešení v rámci fotografování se může jevit pořizovací cena manipulátoru oproti konvenčním stabilizátorům, ale i ta oproti minulým letům poklesla a můžeme v budoucnu očekávat, že se sortiment cenově „dostupných“ (myšlena cenová dostupnost pro firmy, ne pro fyzické osoby) robotických manipulátorů bude postupně zvětšovat s narůstající nabídkou a s tím, jak se budou manipulátory rozšiřovat do nových odvětvích.

Diplomová práce je členěna do tří základních částí. Předmětem první kapitoly je popis zařízení, která mají být použita v rámci práce a jejich základní parametry. V rámci druhé kapitoly jsou představeny použité algoritmy pro sledování objektů a jejich implementace spolu s určením nové polohy manipulátoru na základě dat z těchto algoritmů a vazeb v rámci manipulátoru. Třetí kapitola obsahuje stručné shrnutí odvedené práce a zhodnocení dosažených výsledků spolu s nastíněním případných vylepšení a možného pokračování.

1. POUŽITÁ ZAŘÍZENÍ

Tato část obsahuje základní informace o použitém robotickém manipulátoru, použitých kamerách a optických snímačích, které jsou instalovány na manipulátoru. Jelikož původní myšlenka byla, že by se mohlo jednat o rozšiřující aplikaci pro projekt RoScan, jedná se tedy o zařízení používaná v rámci tohoto projektu bez možnosti změny nebo přidání dalších snímačů. Tento fakt přímo ovlivňuje principy, které je možné v rámci této práce použít.

1.1 Obecné informace o robotických manipulátorech (průmyslových robotech)

Definice pojmu průmyslový robot není v rámci vědecké obce zcela jednotná a jednoznačná. Průmyslový robot je dle mezinárodní normy ISO 8373:2012 (Robots and robotic devices — Vocabulary) definován jako „Automaticky ovládaný, reprogramovatelný, víceúčelový manipulátor , jež je programovatelný ve třech nebo více osách, které mohou být buď stacionární nebo mobilní pro použití v aplikacích průmyslové automatizace“[3], kdy manipulátor je definován jako „Stroj, jehož mechanismus obvykle sestává z řady segmentů vzájemně spojených nebo vůči sobě posuvných za účelem uchopení a / nebo pohybu předmětů (kusů nebo nástrojů) obvykle v několika stupních volnosti“.

Účelem robotických manipulátorů je primárně nahradit člověka v některých činnostech. Z této substituce vyplývá řada výhod jako vyšší přesnost i rychlost manipulátorů, které jsou zároveň i důvodem pro využití robotických manipulátorů, ty jsou využívány například k činnostem, které jsou pro člověka nebezpečné.

Robotický manipulátor by tedy měl být schopen fyzicky ovlivňovat prostředí, ve kterém se nachází a také se v něm pohybovat. Musí být také nějakým způsobem schopen reagovat na prostředí a jeho změny. [4][5][6]

Aby robot splňoval všechny požadavky, které jsou na něj kladeny v rámci definice, je nutné, aby sestával z následujících částí[4][5][6]:

- **Kinematický řetězec** (manipulátor) – jedná se o soustavu na jedné straně pevně spojenou s rámem nebo dalším systémem a s přírubou pro uchycení pracovního nástroje na straně druhé. Je to vlastní tělo robotu, pohyblivá část robotu polohovatelná do různých poloh uvnitř pracovního prostoru. Kinematický řetězec se skládá z:
    - *Počátek souřadné soustavy* (base-point, axes-origin) – místo uchycení robotu ke konstrukci, je to počátek kartézského souřadného systému, který je použit k popisu bodů uvnitř pracovního prostoru robotu.
    - *Ramena* (link, arm) – tuhé spoje různých délek, zakončené na obou koncích pohyblivými klouby. Vybraný materiál ovlivňuje přesnost

pozicování, jejich délka ovlivňuje velikost pracovního prostoru robotu.

- *Klouby* (joint, axis) – pohyblivé spoje ramen, nejčastěji jsou buď rotačního nebo posuvného charakteru a umožňují polohování robotu. Jejich počet, vzájemné umístění a rozsah jejich pohybu určují dostupnost jednotlivých pracovních bodů. Jednotlivé klouby jsou vybaveny motory a enkodéry zajišťující zpětnou vazbu jejich polohy, díky čemuž je možné dosáhnout přesného polohování. Regulace polohy kloubu na žádanou hodnotu se provádí pomocí polohového servomechanismu.
- *Koncový bod* (zápěstí, end-point, tool-point, tool-end) – zakončení kinematického řetězce, který je vybavený nástavcem pro umístění pracovního nástroje. Žádanou pozicí robotu je myšlena poloha tohoto bodu v prostoru.

- **Řídicí jednotka** – mozek robotu, nejčastěji se jedná o průmyslové PC nebo PLC. Většinou bývá oddělena od kinematického řetězce, aby nedošlo k jejímu poškození nebo kvůli lepší dostupnosti. S kinematickým řetězcem je pak spojena vodiči řídícími kloubové motory a snímajícími polohu kloubů. Jednotka mívá rozhraním pro programování činnosti, pro připojení snímačů a pro připojení bezpečnostních prvků. Může disponovat konektory pro připojení programovacího modulu pro přímé učení, diagnostickými moduly apod.

Robotické manipulátory se mohou dělit na základě kinematické koncepce kinematického řetězce, která představuje vzájemné uspořádání pevných ramen a kloubů. Názvy koncepcí jsou odvozeny od známých souřadnicových systémů, ve kterých se pohybuje koncový bod manipulátoru. Mezi základní kinematické koncepce patří [4][5][6]:

- **Robotické rameno** (Articulated) –v dnešní době patří mezi nejpoužívanější. Obsahuje pouze rotační klouby, nejvíce flexibilní, má obvykle největší pracovní prostor, ale nejméně přesné.
- **Kartézská** (Cartesian, Gantry, Linear) – skládá se pouze z posuvných kloubů, zajištuje precizní manipulaci, umožňuje pohyb uvnitř krychle.
- **SCARA** –obsahuje dva rotační a jeden posuvný kloub, je vhodná pro přenášení objektů, výhodou je přesnost a rychlost.
- **Cylindrická** (Cylindrical) – sestává z posuvných kloubů a základna je spojená přes rotační kloub. Polohuje v cylindrickém souřadném systému.
- **Sférická** (Polar) – základna je spojena rotačním kloubem a zbytek těla je kombinací rotačních a posuvných kloubů. Koncový bod se pohybuje ve sférickém pracovním prostoru.

V současné době je na trhu velké množství manipulátorů s různými parametry. Abychom byli schopni udělat správné rozhodnutí a vybrat vhodný typ pro konkrétní činnost, je nutné znát jejich základní parametry a „vyznat se nich“. I přes to, že hlavním rozhodovacím faktorem je na samotném konci cena, tak je důležité vědět, co jednotlivé parametry znamenají, abychom mohli nalézt optimální řešení mezi cenou robotu a jeho parametry. Základními a nejdůležitějšími parametry jsou [4][5][6]:

- **Počet stupňů volnosti** (Degrees of freedom, DOF, Number of axes) – u většiny případů odpovídá počet kloubů kinematického řetězce počtu stupňů volnosti a ty určují, kolik souřadnic v poloze koncového bodu jsme schopni nastavovat. Podmínka pro určitý počet stupňů volnosti je mít stejný nebo vetší počet kloubů, to záleží na použitých typech a vzájemném uspořádání. Pomocí 2 DOF jsme tedy schopni popsat polohu bodu v rovině a pomocí 3 DOF polohu bodu v prostoru. Pro úplný popis polohy pak potřebujeme 6 DOF, to znamená 3 souřadnice pro popis bodu v prostoru a 3 souřadnice pro jednoznačné určení jeho natočení. I když máme robota s 6 DOF, neznamená to automaticky, že jsme schopni dosáhnout všech bodů v určitém prostoru, neboť ještě záleží na vzájemném uspořádání kloubů a ramen.
- **Pracovní prostor** (Working Envelope, Operational Area) – oblast, do které jsme schopni umístit koncový bod manipulátoru. Závisí na kinematické koncepci, počtu stupňů volnosti a rozměrech ramen.
- **Rychlost** (Speed) **a Zrychlení** (Acceleration) – maximální rychlost nebo zrychlení, kterých jsou klouby schopny dosáhnout. Vyšších rychlostí je často dosahováno na úkor nižší přesnosti pohybu. Některé manipulátory proto mohou umožňovat více pracovních módů –rychlejší méně přesné a pomalejší přesnější.
- **Užitečné zatížení** (Carying Capacity, Payload) – hmotnost nákladu, který unese koncový bod, řádově v kg, nutné volit s ohledem na nástroj umístěný na koncovém bodě manipulátoru. Stejně jako u rychlosti můžou manipulátory disponovat různými módy dle hmotnosti – méně kg => vyšší přesnost a zrychlení, větším zatížení => přesnost a max. zrychlení klesá.
- **Přesnost** (Accuracy) **a opakovatelnost** (Repeatability) **polohování** –
    - *Přesnost* – je to maximální odchylka skutečné pozice koncového bodu manipulátoru od požadované pozice. Lze ji označit jako absolutní chybu měření, tj. odchylku měřené hodnoty od správné. Přesnost pozicování lze zlepšit zavedením další zpětné vazby nebo kalibrací.
    - *Opakovatelnost* – určuje s jakou přesností je schopen vrátit se do stejné polohy. Je to tedy rozptyl poloh, do kterých se koncový bod dostane při stejné žádané hodnotě. Tato hodnota však nemá žádnou souvislost se skutečnou pozicí a je dána zejména rozlišením enkodérů, tuhostí

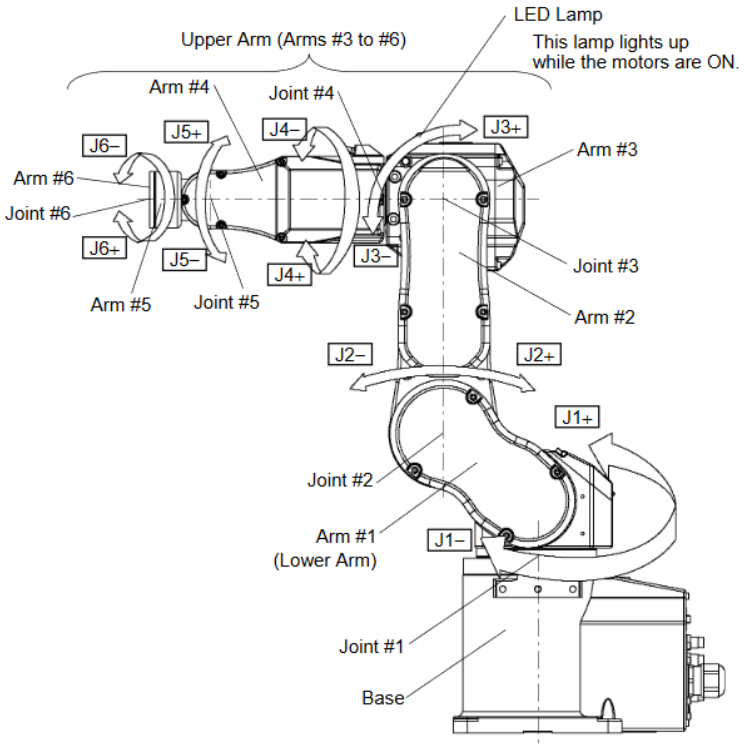
ramen či vůlí v převodech, kterou do značné míry určuje typ převodování.

1.2 Epson C3

V rámci projektu RoScan je použit robotický manipulátor Epson C3. Z toho důvodu bude tento model používán i v rámci této práce. V následující kapitole budou popsány jeho základní vlastnosti.

1.2.1 Popis konstrukce

Epson C3 se řadí k menším, kompaktním a přesným robotickým manipulátorům firmy Epson řady C. Tento model je již v dnešní době nahrazen v nabídce společnosti novějším modelem C4. Model C3 disponuje šesti rotačními klouby(Joint#1-Joint#6), které propůjčují robotu 6 stupňů volnosti, kdy 3 rotační klouby zajišťují pohyb koncového bodu v prostoru prostřednictvím pohybu ramen a zbylé 3 klouby obstarávají natáčení koncového bodu (zápěstí). Robotem lze tedy zcela polohovat v rámci pracovního prostoru kulovitého tvaru. Model C3 se řadí na základě své kinematické koncepce kinematického řetězce mezi robotická ramena. Konstrukce samotného robotu je zobrazena obrázku 1.1.



Obrázek 1.1 Konstrukce manipulátoru Epson C3[1]

Každý kloub je realizován za pomoci střídavého elektrického servopohonu, kdy je jeho moment přenášen na osu pomocí řemenového rozvodu. Převodování zajišťují precizní harmonické převodovky, díky kterým dochází k značné eliminaci vůle v převodech. Jednotlivé klouby jsou opatřeny inkrementálními enkodéry s vysokým rozlišením, které poskytují zpětnou vazbu od polohy jednotlivých kloubů.

Robot je z bezpečnostních důvodů vybaven elektromagnetickými brzdami, které jsou defaultně v zabrzděném stavu a je možné je elektromagneticky odbrzdit. Pokud tedy dojde k výpadku napájení nebo bezpečnostnímu vypnutí, dojde k zabrzdění a robot zůstane v aktuální poloze a nedojde k jeho zhroucení, ohrožení osob, okolního majetku nebo sebe sama.

Na ramenu číslo 4 se nachází signální LED, jejíž sepnutý stav značí, že je napájení motoru aktivní, je dodáván proud motorům. Pokud je však napájení motoru vypnuto nebo se robot nachází ve stavu Emergency Stop je LED vypnuta.

Pohyb robotu je možné programovat v režimu Poin-to-Point, tak i v režimu Continuous Path. Z čehož vyplývá jeho využitelnost v aplikacích s vysokou přesností polohování, či v aplikacích kontinuálních (svařování).[1][6]

1.2.2 Technické specifikace

Následuje souhrn nejdůležitějších technických parametrů z manuálu pro Epson C3.

Přesnost a opakovatelnost

Výrobce udává, že opakovatelnost polohování koncového bodu je ±0,02 *mm*[1]. Z této hodnoty lze vyvozovat, že se jedná o velmi přesného robota. Přesného polohování je dosaženo díky inkrementálnímu enkodéru s velmi vysokým rozlišením na stupeň natočení. Jedná se o desítky tisíc pulzů na stupeň. Přesné hodnoty rozlišení enkodérů a přesnosti polohování pro jednotlivé klouby jsou shrnuty v tabulce 1.1.[1][6]

Tabulka 1.1 Přesnost polohování kloubů manipulátoru Epson C3[1]

Označení kloubu	Přesnost polohování/rozlišení kloubu
Joint #1	0.00000429 °/puls
Joint #2	0.00000429 °/puls
Joint #3	0.00000490 °/puls
Joint #4	0.00000531 °/puls
Joint #5	0.00000524 °/puls
Joint #6	0.00000686 °/puls

Rychlost

Maximální rychlost, jakou může robot vyvinout, záleží na aktuální zátěži a samotném nastavení robotu, jako je jeho mód a podobně. V manuálu uvedené hodnoty říkají, že

pokud uvažujeme zatížení 1 *kg*, je průměrný čas pro cyklus, kdy se robot pohne o 300 *mm* tam a zpět, 0,37 *s*. Maximální úhlové rychlosti, kterých jsou schopny jednotlivé klouby dosáhnout, jsou uvedeny v tabulce 1.2. [1][6]

Tabulka 1.2 Maximální úhlová rychlost kloubů manipulátoru Epson C3[1]

Označení kloubu	**Maximální úhlová rychlost kloubu**
Joint #1	450 °/s
Joint #2	450 °/s
Joint #3	514 °/s
Joint #4	553 °/s
Joint #5	553 °/s
Joint #6	720 °/s

Zrychlení

Řídící jednotka automaticky dopočítává maximální zrychlení (zpomalení) pro každý kloub na základě zadaných parametrů a podle hmotnosti břemene a jeho momentu setrvačnosti. Tím chrání robota před nadměrnými silami (dostředivé, odstředivé …), které by mohly být při pohybu vyvíjeny na klouby a přesáhnout tak povolené meze, tím také optimalizuje pohyb manipulátoru. Povolený moment a moment setrvačnosti pro jednotlivé klouby jsou uvedeny v tabulce 1.3. [1][6]

Tabulka 1.3 Přípustné momenty a momenty setrvačnosti pro Epson C3[1]

Označení kloubu	**Přípustný moment**	**Přípustný moment setrvačnosti ($GD^2/4$)**
Joint #4	4,41 N·m (0,45 kgf·m)	0,15 $kg \cdot m^2$
Joint #5	4,41 N·m (0,45 kgf·m	0,15 $kg \cdot m^2$
Joint #6	2,94 N·m (0,3 kgf·m)	0,1 $kg \cdot m^2$

Parametry motorů

Robot disponuje AC servopohony o výkonu 50-400 *W*, které mají točivý moment 8-10 N.m. Nejsilnější motory se nachází u základny robotu a slabší motory jsou postupně odstupňovány směrem ke koncovému bodu, jelikož motory blíže koncovému bodu mohou být slabší, jelikož musí překonat menší moment síly. [1][6]

Užitečné zatížení

Klasické zatížení je 1 *kg*. Při tomto zatížení jsme schopni dosáhnout maximální rychlosti a zrychlení. Všeobecně je deklarováno maximální zatížení 3 *kg*, avšak v případě, že je omezena orientace pátého kloubu tak, že vždy směřuje kolmo dolů k zemi, je maximální zatížení 5 *kg*. Čím více se přibližujeme k horní hranici zatížení, tím více snižuje řídící

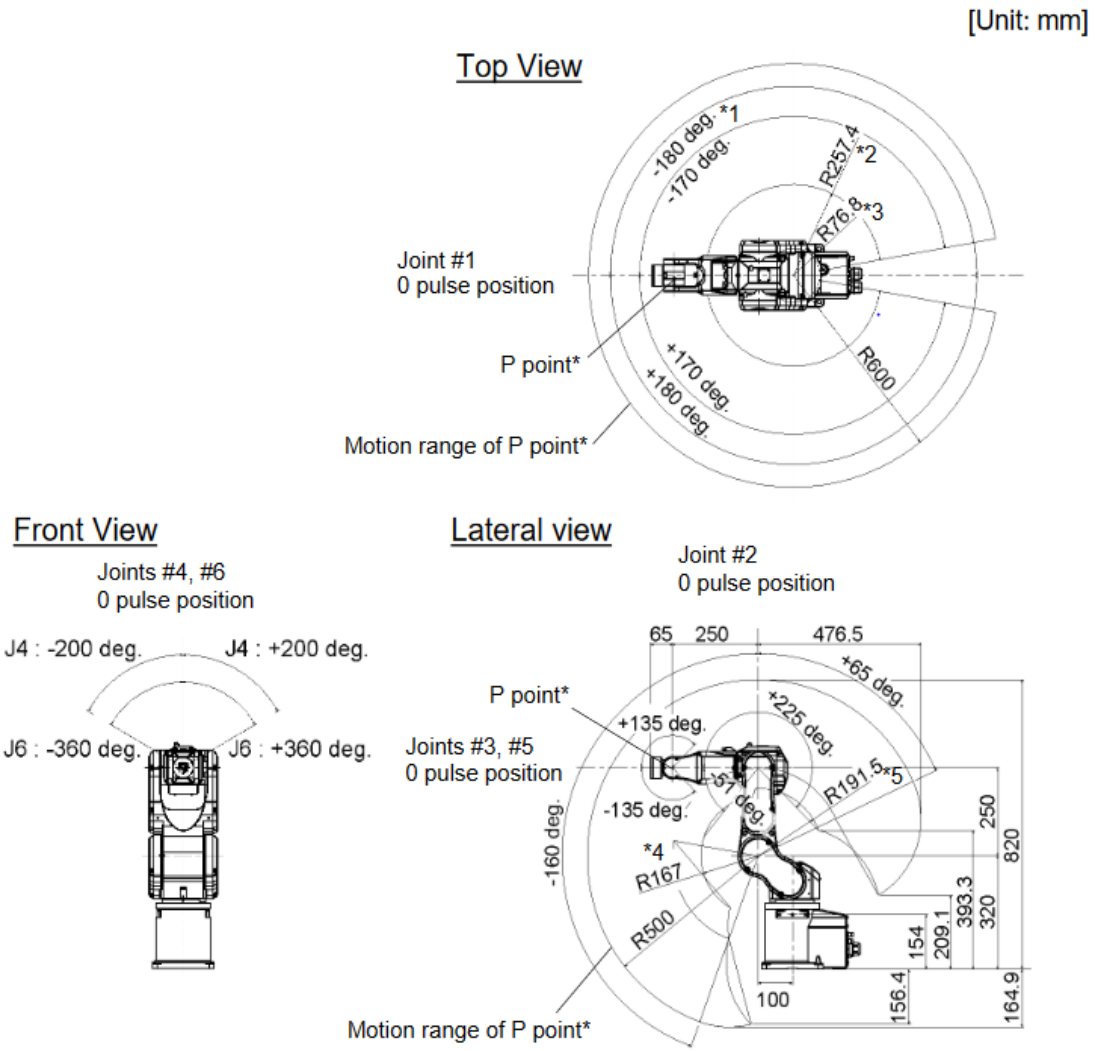
jednotka maximální rychlost a zrychlení, aby zamezila poškozením plynoucích z působení nadměrných sil. [1][6]

Pracovní prostor

Šest kloubů společně s šesti stupni volnosti umožňují robotu Epson C3 polohovat koncový bod v rámci kulového prostoru ve všech šesti pozicích, kterými se dá bod popsat. Klouby mají široký rozsah pohybu, což umožňuje pohyb ve velkém pracovním prostoru, jak je ukázáno na obrázku 1.2. Rozsahy natočení jednotlivých kloubů ve stupních i pulzech jsou shrnuty v tabulce 1.4. Pracovní prostor závisí vedle mezí pohyblivosti kloubů také na způsobu montáže robotu. Epson C3 umožnuje tři základní způsoby uchycení robotu - na stůl, na strop, nebo na stěnu. Ve všech třech případech je počátek souřadné soustavy v místě uchycení, osa Z je kolmá k zemi a osa Y značí, kde se nachází hlavní část pracovního prostoru (kolmo před bází robotu), kdy v tomto směru má robot největší volnost a dosah. V našem řešeném případě je robot uchycen ke stolu. [1][6]

Tabulka 1.4 Maximální rozsahy pohybu kloubů manipulátoru Epson C3[1]

Označení kloubu	**Maximální rozsah ve stupních**	**Maximální rozsah v pulzech**
Joint #1	±170° (±180°)	±4951609 (±5242880)
Joint #2	-160° - +65°	-4660338 - +1893263
Joint #3	-51° - +225°	-1299798 - +5734400
Joint #4	±200°	±4700057
Joint #5	±135°	±3217222
Joint #6	±360°	±6553600



Obrázek 1.2 Rozsah pohybu manipulátoru Epson C3[1]

1.3 Řídící jednotka RC180

V našem případě je manipulátor Epson C3 ovládán pomocí univerzální řídící jednotky RC180. Jedná se o výpočetní jednotku podobnou průmyslovým automatům. Je vybavena 32 bitovým Ultra low voltage procesorem. Disponuje pouze pamětí Flash, která slouží jako paměť programu.

Jednotka může být v základu programována skrze komunikační rozhraní ethernet, USB 2.0 nebo pomocí USB paměťových médií. K robotu je možné připojit také vstupně-výstupní karty pro Fieldbus (Profibus, DeviceNet…) či RS-232. Existuje také možnost programování skrze připojené speciální proprietární zařízení (Teach Pendant), které umožňuje jednoduché programování robotu bez znalosti programovacích jazyků.

Pomocí digitálních vstupně-výstupních karet lze také řídit i okolní technologie, jejichž činnost má spojitost s činností robotu, jako jsou dopravníky, limitní snímače, světelné brány apod. Kvůli bezpečnosti disponuje jednotka i Emergency Portem, jehož funkce není programově upravitelná. Obsahuje signály pro obvody Emergency Stop, koncové bezpečnostní spínače a Reset.

Samotný program pro ovládání robotu je psán ve vlastním skriptovacím jazyce SPEL+, který obsahuje příkazy pro popis trajektorie.

Robota lze řídit také pomocí nadřazeného systému, kdy se nejčastěji používá PC. Ten pak může pomocí dostupných komunikačních sběrnic buďto posílat pouze souřadnice, nebo robota řídit celého.

Řídící jednotka provádí mnoho funkcí jako např. postupné provádění programu ve SPEL+, transformuje kartézské souřadnice, se kterými pracujeme v kartézském souřadném systému robotu, do kloubových souřadnic, kontroluje meze pohybu, aby nedošlo ke sebepoškození a detekuje kolize ramene s cizími předměty za pomoci měření zatížení motorů, řeší také optimalizaci dynamiky pohybu. [2][6]

1.4 Kamera

Manipulátor Epson C3 je v rámci projektu RoScan osazen dvěma kamerami, jedná se o termokameru a RGB kameru. RGB kamera má být využita jako hlavní snímač v rámci sledovacího algoritmu. Termokamera je používána pro měření teploty v rámci projektu RoScan, ale vzhledem k jejímu rozlišení nebude v rámci sledování objektů využita.

1.4.1 Termální kamera

Jedná se o termální kameru Xenics GOBI1954. Kamera disponuje rozlišením 385 x 288 pixelů, roztečí pixelů 25 μm a spektrální odezvou pro rozsah vlnových délek 8−14 μm.

1.4.2 Barevná kamera

Jedná se o barevnou kameru ImagingSource DFK51BG02.H. Kamera disponuje rozlišením 1600 x 1200 pixelů. Je vybavena CCD čipem Sony ICX274AQ typu 1/1,8 palce, o velikosti 8,6 x 6,8 mm, s velikostí pixelu 4,4 x 4,4 mikronu. Její maximální frekvence snímání činí 12 FPS. Kamera využívá globální uzávěrku a disponuje možností externího spouštění zachycování se zpožděním menším než 5 µs.[14]

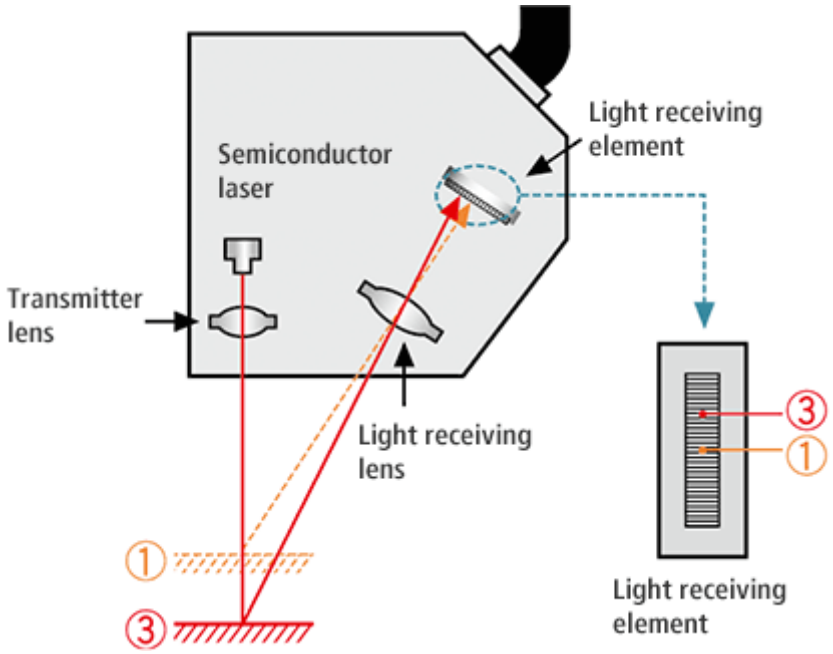
1.5 Laserový skener

1.5.1 Triangulační metoda

Účelem aktivní triangulační metody je výpočet vzdálenosti na základě úhlu odraženého paprsku, který byl snímačem vyzářen na měřený objekt. Schématické znázornění

fungování senzoru využívajícího triangulační princip můžete vidět na obrázku 1.3.

Princip měření vychází ze situace, kdy je paprsek odražen od detekovaného předmětu pod konstantním úhlem, tak je vzdálenost místa dopadu odraženého paprsku na přední stranu senzoru úměrná vzdálenosti detekovaného předmětu od čela senzoru. V praxi tedy dochází k vyhodnocování místa dopadu odraženého paprsku. To má za následek spolehlivější detekci, odolnější proti rušení, než mají snímače pracující na principu vyhodnocování intenzity dopadajícího paprsku. Přesnost je především dána schopností optického snímače senzoru detekovat místo dopadu odraženého paprsku o jakékoliv větší intenzitě, než je minimální detekovatelná intenzita.



Obrázek 1.3 Fungování senzoru na bázi triangulačního principu [9]

Konstrukce senzoru obvykle sestává z laserové diody, která vysílá paprsek viditelného světla směrem k měřenému objektu. Odražený paprsek pak dopadá na vstupní optický systém, díky kterému je paprsek nasměrován na plochu světlocitlivého snímače. Nejčastěji se používají světlocitlivé čipy typu CCD nebo CMOS.

Některé senzory, které fungují na principu triangulace, jsou schopny i měření tloušťky průhledných či průsvitných homogenních předmětů a ploch. V takovém případě jsou na přijímači detekovány dvě dopadající světelné stopy, jedna tvořená částí odraženého světla od vrchní vrstvy a druhá tvořená částí odraženého světla od druhé vrstvy (může být i podklad).

Vzhledem k omezeným rozměrům přijímacího čipu je i měřící rozsah limitován. Maximální hodnoty měřené vzdálenosti bývají v řádu milimetrů až metrů. Nevýhodou je, že s rostoucím měřícím rozsahem měření se snižuje hodnota linearity a rozlišení, čím větší je vzdálenost, tím menší je rozlišení. Proto se můžeme setkávat s tím, že jsou

uváděny v parametrech dvě či více hodnot rozlišení pro různé rozsahy vzdáleností měřícího rozsahu. Rozlišení je dáno nastavením optiky přijímače, ale především tím, že stejná změna vzdálenosti generuje blíže ke snímači větší změnu úhlu odraženého paprsku, než ve větší vzdálenosti od snímače, přičemž samotné rozlišení světlocitlivého snímače je pevně definované.[7][8]

1.5.2 Skener MicroEpsilon ScanCONTROL

Projekt RoScan využívá pro získávání dat za účelem 3D generování povrchu kompaktní laserový profilový skener s kontrolérem integrovaným ve snímači. Jedná se o model MicroEpsilon ScanCONTROL2900-100/BL. Tento snímač funguje na bázi triangulačního principu. V rámci této práce je však použit pouze k měření vzdálenosti od sledovaného objektu v rámci jeho měřícího rozsahu.

Parametry

Přesnost a rozlišení

Rozlišení je závislé na materiálu měřeného objektu. Výrobce uvádí hodnoty rozlišení 12 μm s maximální odchylkou ±0,10 %. Hodnoty přesností jsou uváděny jako referenční pro Micro-Epson standardní objekt, kovový, difuzně odrážející materiál. Chyba linearity je pro celý standardní rozsah ±0,012 %. Hodnota opakovatelnosti není zvlášť uvedena, takže můžeme předpokládat, že se vyskytuje v intervalu rozlišení. Rozlišení v ose X je 1280 bodů na šířku profilu.[10]

Oblast měření

Skener má dva rozsahy - standardní a prodloužený. Standardní rozsah činí 100 *mm,* kdy nejmenší povolená vzdálenost měřeného objektu od senzoru je 190 *mm* a největší měřitelná vzdálenost objektu od senzoru je 290 *mm*. Šířka snímaného profilu se v tomto případě pohybuje od 83,1 *mm* do 120,8 *mm*. Prodloužený rozsah je 265 *mm*, kdy nejmenší povolená vzdálenost měřeného objektu od senzoru je 125 *mm* a největší měřitelná vzdálenost objektu od senzoru je 390 *mm*. Šířka snímaného profilu se v tomto případě pohybuje od 58.5 *mm* do 143.5 *mm*. Zorné pole svírá úhel 21,4°.[10]

Frekvence skenování

Skener je schopen měřit při vysoké frekvenci měření a to až 2000 *Hz*. Při nejvyšší rychlosti se jedná o velké množství (2560000 bodů za sekundu) dat, které pak může být náročné na zpracování. Avšak ve standartním režimu se jeho frekvence snímání profilů pohybuje v rozmezí do 300 *Hz*. [10]

Úhlové rozlišení

Počet měřících bodů na šířku profilu činí 1280px. Vezmeme-li v potaz úhel zorného pole dostaneme úhlové rozlišení přibližně $1{,}67 \cdot 10^{-2}$ stupně. [10]

Zdroj světla
Zkratka /BL v kódovém označení modelu laseru znamená, že skener disponuje modrým laserem pro polopropustné až propustné, červeně žhnoucí a organické materiály. Jedná se o polovodičový laser třídy 2M s vlnovou délkou 405 *nm* a výkonem do 8 *mW*. [10]

Komunikační rozhraní
Laserový skener je vybaven rozhraními Ethernet GigE Vison (Ethernet), RS422(poloviční duplex) a klasickými digitálními vstupy. Senzor může být konfigurován z PC přes Ethernet nebo RS422. Vzhledem k vysokým datovým tokům je vhodné volit rozhraní, které dokáže přenášet informace o skenovaném profilu pro celý rozsah frekvencí, tedy Ethernet. Programovatelné rozhraní RS422 může být použito jako trigger nebo pro synchronizaci. Výstupní změřené hodnoty mohou být vyčítány pomocí řady průmyslových komunikačních protokolů, jako jsou Ethernet (UDP / Modbus TCP), RS422 (ASCII / Modbus RTU), PROFINET, EtherCAT, EtherNet/IP. [10]

1.6 Softwarové nástroje

V této kapitole jsou zmíněny základní informace o softwarových prostředcích, které byly využity v rámci této práce. Jedná se převážně o knihovnu OpenCV.

1.6.1 OpenCV

OpenCV neboli Open Source Computer Vision Library je knihovna programovacích funkcí převážně se zaměřující na počítačové vidění a zpracování obrazu v reálné čase. Obsahuje stovky algoritmů pro počítačové vidění.

OpenCV je psána v jazyce C++, jejím primárním rozhraním je tedy taktéž C++, existují však i verze rozhraní pro Python a Javu. Knihovna je podporována na operačních systémech Linux, MacOS, Windows, iOS a Android.

Původně byla vyvíjena společností Intel. Jedná se o multiplatformní knihovnu. Je zdarma k užívání pod licencí BSD 3-Clause License. Je tedy možné ji zdarma používat i ke komerčním účelům.

Veškeré počítačové vidění, které jsem v rámci této práce programoval, bylo naprogramováno za pomocí funkcí poskytovaných touto knihovnou, respektive jejím portem do jazyka C#.

EmguCV
Jedná se o multiplatformní .Net wrapper, přidává rozhraní .NET, knihovny OpenCV. Umožňuje tedy volání funkcí z knihovny OpenCV i z jazyků kompatibilních s .NET, tedy i C#. Je možné ho zkompilovat ve Visual Studiu a tuto možnost jsem využil i v rámci této práce. Je podporován na platformách stejně jako OpenCV na Windows, Linux, Mac OS,

iOS a Android.

Vzhledem k tomu, že byl projekt RoScan doposud programován v jazyce C#, proto v rámci své práce používám jazyk C# a tento wrapper, aby mohla být má práce lépe začlenitelná do projektu.

Hardwarová akcelerace

Knihovna OpenCV poskytuje v současné době také možnost zrychlení provádění algoritmů pro zpracování obrazu za pomoci grafického procesoru. Pro tuto problematiku je možné využít dva přístupy. OpenCV obsahuje model CUDA, jehož algoritmy spolupracují s procesory firmy NVIDIA a jsou na nich urychlovány. Druhou možností je urychlování za pomocí OpenCL (Open Computing Language), jedná se o průmyslový standard pro paralelní programování heterogenních počítačových systémů. Díky využití funkcionalit OpenCL, kdy některé algoritmy v rámci OpenCV mají i OpenCL implementaci, je možné dosáhnout přístupu k GPU za účelem provádění negrafických výpočtů (GPGPU). Hardwarové zrychlení nám může přinést zrychlení v řádů stovek procent oproti počítání na běžném CPU. Mnoho algoritmů počítačového vidění totiž dokáže pracovat na GPU mnohem efektivněji.

2. PRAKTICKÉ PROVEDENÍ

V následující kapitole se můžete dočíst o algoritmech, které jsem použil pro sledování objektů, ať se jedná o algoritmy implementované v knihovnách 3. stran, nebo vlastní algoritmy založené na detekci vybraného objektu v každém snímku sekvence. Možnost volby algoritmů byla značně omezena v závislosti na tom, že nemohlo být měněno rozložení, počet ani typ snímačů a byla požadována nezávislost sledovacího algoritmu na typu/třídě sledovaného objektu. To hned na začátku vyřadilo některé principy, jako jsou neuronové sítě a stereovizi (pasivní triangulace) z úvahy, kvůli nevhodnosti nebo neproveditelnosti v závislosti na zadání.

2.1 Sledování vs detekce

Pro semestrální práci jsem zvolil dva způsoby, jak detekovat pozici sledovaného objektu v obraze. Jde o trackovací algoritmy, které jsou dostupné v rámci knihovny OpenCV a druhým způsobem je „obyčejné“ detekování pozice objektu v obrazu na základě porovnávání descriptorů – feature matching. Než v dalších podkapitolách popíšu použité algoritmy, vysvětlím nejdříve na tomto místě hlavní rozdíly trackovacích algoritmů oproti klasické detekci. Je nutné mít na paměti, že to, co zde bude uvedeno, nemusí platit pro každý trackovací algoritmus. [11]

- **Trackovací algoritmy bývají rychlejší** – Ve většině případů můžeme říct, že trackovací algoritmy jsou rychlejší. Představme si, že sledujeme nějaký objekt, který jsme detekovali na posledním snímku. Máme tedy dost informací o jeho podobě, navíc známe jeho polohu na předcházejícím snímku, a ze znalosti místa výskytu na předchozích snímcích můžeme určit i směr a rychlost pohybu. Proto na dalším snímku můžeme využít tyto informace a předvídat pozici sledovaného objektu na tomto snímku a zmenšit tak oblast hledání na okolí kolem předpokládaného místa výskytu. Trackovací algoritmy využívají pro nalezení hledaného objektu všechny zmíněné informace, zatímco algoritmy pro klasickou detekci pracují pouze s informacemi z aktuálního snímku a prohledávají ho celý.
- **Mohou fungovat tam, kde detekce selhává** – Pokud nastane situace, že dojde k nějaké úrovni překrytí sledovaného objektu jiným objektem, je pravděpodobné, že detekce může selhat. Na druhou stranu je dobrý sledovací algoritmus schopen zvládnout sledovat předmět i při určité míře překrytí. Trackovací algoritmy však mohou selhávat, pokud se sledovaný objekt přesune za překážku po delší časový úsek nebo pokud se objekt pohybuje moc rychle a algoritmus nedokáže tento pohyb zachytávat.
- **Trackovací algoritmy zachovávají identitu objektů** – V případě klasické detekce, je výsledkem informace o výskytu daného typu objektu v obraze, například ve formě čtyřúhelníku, ve kterém se daný objekt nachází. Neexistuje tedy žádná přímá

spojitost mezi detekovanými objekty na jednotlivých snímcích. Na rozdíl od toho trackovací algoritmy rozeznávají od sebe jednotlivé detekované instance a dokážou tak určit vazby mezi detekovanými objekty na předchozím snímku a detekovanými objekty na aktuálním snímku. Dokážou určit trasu pohybu jednotlivých detekovaných objektů a odlišit je. To má vliv především při sledování více stejných objektů zároveň.

2.2 Sledovací algoritmy v rámci OpenCV

Jedná se o velmi komplexní algoritmy, kdy není účelem této práce je zde rozepisovat a komplexně vysvětlovat, neboť mohou být i značně komplikované a samotné téma těchto aloritmů by mohlo být předmětem další individuální práce. U konkrétních algoritmů, které byly vybrány a testovány, bude v této práci pouze krátce zmíněno, na jakém principu pracují a jaké byly doposud zjištěny jejich silné a slabé stránky, a ve kterých situacích by bylo vhodné daný algoritmus využít.

Obrázek 2.1 Ukázka výsledků sledování objektu za pomocí trackovacích algoritmů

2.2.1 Implementace

Způsob implementace trackovacích algoritmů je na rozdíl od samotných algoritmů poměrně jednoduchý. Jelikož se jedná o algoritmy, které si ve své třídě uchovávají informace o minulých výskytech a podobách sledovaného objektu, tak existují dva způsoby jak náš algoritmus, který je využívá, implementovat. Prvním způsobem by bylo vytvoření metody/funkce, kdy by instance třídy trackeru byla vstupním parametrem a zároveň i výstupem. Tímto způsobem dostaneme nazpět kromě místa výskytu sledovaného objektu i aktualizovaný tracker. Druhým způsobem by bylo vytvořit vlastní třídu, kdy by v rámci metod byl–používán člen dané třídy, který by byl instancí třídy trackeru. V této práci jsme zvolili druhý přístup.

Před zahájením sledování objektu pomocí trackeru je vždy nutné jej inicializovat. K tomuto účelu se nachází v rámci naší třídy metoda pro inicializaci initialize. Jediným argumentem této metody je snímek, ze kterého chceme vybrat sledovaný objekt. Po zavolání této metody se zobrazí okno s vstupním obrazem a po uživateli je požadován výběr oblasti s hledaným objektem. Výběr je prováděn kliknutím a tažení pomocí kurzoru myši, kdy tak vybereme obdélník, ve kterém se nás předmět zájmu nachází. Potvrzení oblasti výběru provedeme stiskem klávesy „space“ nebo „enter“. Výsledkem této operace jsou souřadnice a rozměry obdélníku, který je použit pro inicializaci trackeru spolu s daným vstupním obrazem. Pokud proběhla inicializace úspěšně, navrátí metoda jako návratovou proměnnou hodnotu true, jinak false. Třídy trackerů neumožnují reinicializaci, neboť je zakázaná na úrovni knihovny, pokud byla třída již inicializována, a my ji chceme reinicializovat, musíme nejdřív původní třídu zničit a založit zcela novou instanci dané třídy. Je důrazně doporučeno používat pro inicializaci snímek scény těsně předtím, než začneme se sledováním, jelikož je pozice vybraného obdélníku použitého pro inicializaci použita jako výchozí místo pro hledání sledovaného objektu. Pokud by se pozice objektu na inicializačním snímku značně lišila od pozice na prvním snímku sledovací sekvence, mohlo by dojít k selhání algoritmu hned na začátku.

Máme-li tracker inicializován, můžeme přejít k metodě sledování. Implementace sledování je již jednoduchá, jediné, co potřebujeme, je aktuální snímek scény, ve které se nachází sledovaný objekt. Tento snímek převedeme do maticové podoby a předáme jako argument metodě Update trackeru. Metoda při úspěšném nalezení objektu vrací souřadnice a rozměry obdélníku, ve kterém se má nacházet sledovaný objekt. Zda-li byl objekt nalezen, se dozvíme pomocí návratové hodnoty metody, kdy *true* značí úspěch a *false* neúspěch.

Nastavení trackerů

Parametry trackerů byly ve výsledku ponechány na defaultních hodnotách vzhledem ke skutečnosti, že jsou to obecně doporučené hodnoty vyplývající z vědeckých článků, na jejichž základě byly tyto algoritmy do knihovny OpenCV implementovány.

Druhým a nejspíše hlavním důvodem je, že i když proběhly pokusy změnit parametry trackerů dle uvážení, nedokázali jsme dosáhnout prokazatelně lepších výsledků a ve většině případů došlo ke zhoršení sledovacích schopností trackerů. K nedosažení žádaného výstupu mohlo dojít na základě nedostatečného pochopení principů skrývajících se za jednotlivými parametry, z důvodu složitosti a komplexnosti jejich algoritmů. Proto byl zvolen jiný přístup k problému a to ten, že byly vyhledávány články, či studie k tématu těchto sledovacích algoritmů za účelem nalezení informací popisujících vlivy jednotlivých parametrů na výkon a chování algoritmu, popřípadě nalezení informací poskytujících instrukce k nastavení těchto sledovacích algoritmů. Z publikovaných článků k této problematice byla daná oblast prostudována, avšak úroveň získaných znalostí nepostačovala k vyřešení tohoto problému. Navíc nalezené články většinou neřešily problematiku vlivu nastavení jednotlivých parametrů, ale různé jejich modifikace a porovnávání výkonnosti těchto modifikovaných algoritmů. Výsledek byl tedy takový, že nebyl nalezen princip, na jakém bychom měnili parametry tak, abychom dosáhli požadovaného výsledku. Proto byly ponechány defaultní hodnoty knihovny.

2.2.2 Použité trackery a jejich vlastnosti

Na tomto místě jsou zmíněny použité trackovací algoritmy, které byly testovány, a především jejich silné a slabé stránky. Algoritmy jsou popsány velmi krátce, bude zmíněn pouze obecný princip, z důvodu jejich značné složitosti a komplexnosti. Nejprve však budou zmíněny některé principy sledování objektů, které jsou pro pochopení algoritmů nezbytné.

Jelikož v rámci algoritmů sledování jsme sledován objekt na sekvenci snímku až do aktuálního snímku, známe tedy, jak se pohyboval. Jinak řečeno, víme parametry modelu pohybu. Ten zahrnuje znalost polohy a rychlosti (s jejím směrem) v předchozích snímcích. I kdybychom nevěděli nic jiného o objektu, přesto bychom mohli odhadnout novou pozici objektu na základě modelu pohybu a pravděpodobně bychom se přiblížili blízko místu, kde je nová poloha objektu i ve skutečnosti.

Kromě informací o pohybu také víme, jak objekt vypadal na předchozích snímcích. Na základě těchto dat můžeme sestavit model vzhledu, který uchovává informaci o vzhledu objektu. Tohoto modelu využijeme, abychom prohledali blízké okolí místa, které bylo odhadnuto za pomoci modelu pohybu a můžeme tak zpřesnit odhad polohy objektu.

V ideálním případě se vzhled předmětu s časem příliš nemění, stačilo by tedy mít šablonu jako model vzhledu, kterou bychom porovnávali. Ale vzhledem k situaci, kdy ve skutečném světě se může vzhled z pohledu obrazových informací z kamery velmi změnit,

zavedly některé trackovací algoritmy modely vzhledu jako klasifikátory, které jsou trénované online za účelem aktualizace modelu vzhledu.

Klasifikátor má za úkol rozhodnout, zda je obdélníková oblast obrazu objektem nebo pozadím. Klasifikátor má jako vstup část obrazu a vrátí hodnotu od 0 do 1, která představuje pravděpodobnost, že tato část obrazu obsahuje objekt. Hodnota 0 je absolutní jistota, že se jedná o pozadí, a hodnota 1 je jistota, že se jedná o objekt.

Klasifikátor je učen online, tedy za chodu programu. Trénování probíhá tak, že jsou klasifikátoru předkládány pozitivní (v našem případě objekt) a negativní příklady (v našem případě pozadí), na kterých se trénuje. [11]

MIL

Algoritmus MIL (Multiple Instance Learning) je právě jedním ze sledovacích algoritmů založených na online učení. Jeho klasifikátor tedy potřebuje být učen za chodu programu předkládáním pozitivních a negativních příkladů objektů. Avšak místo, aby jako pozitivní příklad objektu používal algoritmus MIL jen současnou pozici objektu, vezme navíc v potaz i malé okolí kolem aktuální pozice a z něj vygeneruje další potencionálně pozitivní prvky.

S faktem, že takto vygenerované příklady nejsou zcela vystředěné, se vypořádává MIL tak, že v jeho případě nespecifikujeme pozitivní a negativní příklady, ale pozitivní a negativní skupiny. Sbírku obrázků v pozitivní skupině netvoří jen pozitivní příklady. Ve skutečnosti musí být pozitivní pouze jeden obrázek ve skupině. V prvním kroku máme tedy skupinu tvořenou výřezem vycentrovaným na aktuální polohu a výřezy v blízkém okolí tohoto bodu. I přesto, že by byla v dalších krocích aktuálně určená poloha sledovaného objektu lehce nepřesná, máme dobrou pravděpodobnost, že alespoň jeden z příkladů z blízkého okolí, které jsou umístěny do pozitivní skupiny vzorků, je hezky vycentrovaný obrázek sledovaného objektu. [11] [15]

Během testování podával tracker MIL dobré výsledky z hlediska přesnosti a spolehlivosti. Co se týče rychlosti, patří mezi pomalejší trackery. Avšak docházelo u něj k falešně pozitivním detekcím a selhával při obnovení sledování z velkého nebo úplného překryvu.

KCF

KCF v tomto případě znamená Kernelized Correlation Filters (kernelizované korelační filtry). Je založen na podobném principu jako MIL popsaný výše. Na rozdíl od něj však využívá skutečnost, že více pozitivních vzorků v pozitivní skupině má v případě MIL trackeru velké překrývající se oblasti. Tato překrývající se data vedou v případě KCF k příhodným matematickým vlastnostem, které jsou využívány KCF trackerem k tomu, aby bylo sledování přesnější i rychlejší. [11] [16]

Tracker KCF během testování měl srovnatelné nebo lepší výsledky něž předešlý tracker, to se dalo očekávat na základě popisu algoritmu, na kterém je tracker KCF

postaven. Dosahoval i vyšších rychlostí než většina ostatních trackerů (až na MOSSE). Nejlépe však odhalil situaci, kdy nedokázal detekovat objekt, jevil se tedy jako tracker s nejmenším množstvím falešně pozitivních detekcí. Jako většina algoritmů má však problém při obnově po úplném překrytí objektu.

CSRT

Tento sledovací algoritmus byl implementován do OpenCV na základě práce A. Lukezic a jeho kolegů o Rozlišujícím Korelačním Filtru s Kanálovou a Prostorovou Spolehlivostí (Discriminative Correlation Filter with Channel and Spatial Reliability, CSR-DCF) , který navrhli. Přizpůsobili omezující korelační filtr mapou prostorové spolehlivosti a odhadovanou spolehlivost kanálu s kanálem omezujících funkcí. V jejich navrhovaném algoritmu je objekt lokalizován součtem odezev naučených korelačních filtrů a váhováním odhadovanou hodnotou spolehlivosti kanálu. Tato navrhovaná metoda implementace CSR-DCF byla v knihovně OpenCV provedena jako CSRT tracker. [11] [17]

Soudě podle pokusů, které byly provedeny v rámci výběru trackerů pro naši aplikaci, se jedná o nejpřesnější tracker mezi testovanými. Na druhou stranu se řadí vedle MIL mezi pomalejší algoritmy. Stejně jako většina testovaných trackerů má také problém při obnově po úplném překrytí objektu.

MOSSE

Minimum Output Sum of Squared Error (MOSSE), tedy Minimální výstupní součet kvadrátů chyb, používá pro sledování objektů adaptivní korelaci, která při inicializaci pomocí jednoho snímku vytváří stabilní korelační filtry. MOSSE tracker má být odolný vůči změnám v osvětlení, měřítku, póze a netuhých deformacích. Dokáže detekovat také překrytí na základě poměru vrcholu-k-bočnímu laloku (sidelobe), což umožňuje trackeru pozastavit a pokračovat tam, kde přestal, když se objekt znovu objeví. Je poměrně přesný jako jiné komplexní sledovače a mnohem rychlejší. [11] [18]

Při mém testování jsem zjistil, že MOSSE tracker pracuje při vyšších fps, je řádově rychlejší než ostatní testované algoritmy. Avšak v měřítku přesnosti/výkonu zaostává za některými testovanými sledovači založenými na hlubokém učení. Vypořádává se celkem dobře s překrytím sledovaného objektu a je schopen se z něj vzpamatovat, pokud se objekt začne pohybovat zpět v místě, kde ho tracker ztratil. I přesto je jeho největší předností rychlost.

TLD

TLD znamená Tracking, Learning, and Detection, tedy sledování, učení a detekci. Jak název naznačuje, tento tracker rozděluje úkol dlouhodobého sledování na tři komponenty

- sledování , učení a detekci. Tracker tedy sleduje objekt od snímku ke snímku. Detektor lokalizuje všechny dosud pozorované podoby a v případě potřeby koriguje tracker. Učení odhaduje chyby detektoru a aktualizuje ho, aby se zamezilo těmto chybám do budoucna. [11] [19]

Během testování jsem objevil, že výstup, indikující pozici sledovaného objektu tohoto trackeru, má tendenci občas skákat a produkovat tak falešně pozitivní nálezy. Pokud například sledujeme objekt a na scéně jsou další podobné objekty, může tento tracker někdy dočasně skočit od sledování vybraného objektu a sledovat jiný objekt, než který jsme původně ke sledování vybrali, a to i v oblasti, která nebyla v blízkosti posledního výskytu sledovaného objektu. Jako jediná výhoda se jeví fakt, že tento tracker sleduje pohyb ve velkém rozsahu pohybu a dobře se vypořádává s překrytím. Je to také jediný tracker, který dokázal navázat na sledování objektu poté, co se objekt vynořil na druhé straně překážky.

Shrnutí

Pokud bychom potřebovali co nejvyšší rychlost, bylo by vhodné zvolit MOSSE, jelikož rychlost je jeho největší předností. V našem případě jsme však limitováni rychlostí námi používané kamery, a proto bylo usouzeno, že bychom tedy nedokázali plně využít potenciál rychlostí, na kterých může MOOSE pracovat. Na základě výše uvedeného se tedy pro nás MOSSE nejevil jako správná volba.

Pokud by bylo zapotřebí z nějakého důvodu sledovat objekty s velkými, či úplnými překrytími, jevil se pro tento účel jako nejlepší algoritmus TLD. Ten však vytváří tak velké množství falešně pozitivních detekcí, že by byl pro nás jen velmi těžko použitelný, spíše až nepoužitelný. Navíc v rámci naší aplikace nebyla předpokládána nutnost vypořádat se se situacemi, kdy by došlo k úplnému překrytí sledovaného objektu.

Algoritmus MIL má dobré výkony jak v rychlosti, tak i přesnosti a spolehlivosti detekce sledovaného objektu. Algoritmus KCF ovšem předvádí obdobné a v některých oblastech lepší výsledky. Především má lepší výsledky v detekování falešně pozitivních pozic sledovaného objektu, má tedy méně falešně pozitivních detekcí. Z tohoto ohledu se tedy jedná o nejlepší volbu v případě, že hledáme kompromis mezi rychlostí a přesností.

Nejpřesnějším, ale zároveň i jedním z nejpomalejších algoritmů v našem výběru, je algoritmus CSRT. Jedná se tedy o správnou volbu v případě, že nepotřebujeme rychlost, ale chceme co největší přesnost a spolehlivost.

V konečném výsledku nám tedy vyšli jakožto nejlepší kandidáti pro naše použití trackovací algoritmy KCF a CSRT. Tyto algoritmy byly implementovány jako jedna ze sledovacích metod ve finální verzi práce.

2.3 Sledování objektu pomocí detekce - Feature Matching

V rámci této metody pro sledování objektu v obraze byly využity takzvané Feature detectors, což jsou algoritmy detekující nějaké výrazné vlastnosti objektů v obraze,

například rohy, hrany a podobně, které jsme na základě obrazových dat schopni nějakým způsobem detekovat.

Jak již bylo zmíněno výše, tak na rozdíl od algoritmů, které jsou určeny výhradně jako sledovací algoritmy, detektory detekují specifický objekt v jednotlivých snímcích nezávisle. Nesledují tedy trajektorii pohybu objektu, ale pouze nachází nejlepší možnou shodu mezi dvěma obrázky.

Pokud chceme tedy použít klasické detektory vlastností pro sledování, může to zjednodušeně vypadat následovně. Máme tedy dva obrázky, první je obrázek hledaného objektu a druhý je aktuální snímek z kamery nebo videa. Detektory jsou použity k tomu, aby detekovaly co nejvíce významných bodů vypovídajících o objektech nacházejících se na obrázcích. Každý takový nalezený bod je popsán svým vlastním desktriptorem. Snažíme se pak nalézt co nejlepší shodu mezi detekovanými významnými body na obrázku s hledaným objektem a detekovanými významnými body na aktuálním snímku. Shoda je vyhodnocována na základě kritéria matematické vzdálenosti počítané na základě deskriptorů jednotlivých významných bodů. Tomuto procesu se říká Feature matching. Objekt může být volně zmenšován/zvětšován a natáčen v rovině snímku a lehce natáčen i v ostatních osách, a přesto být detekován. Takto metoda funguje, ale pouze za předpokladu, že je nalezeno dostatečné množství významných bodů.

2.3.1 Implementace

V rámci feauture matchingu byl implementovány dvě odlišné metody pro nalezení shody mezi obrázkem sledovaného objektu a scénou. Jedná se o metody Brute-Force a FLANN.

Princip Brute-Force porovnávače je jednoduchý. Vezme deskriptor jednoho významného bodu z prvního balíku a porovná je, snaží se je spárovat s každým významným bodem z druhého balíku za pomoci výpočtu vzdálenosti. Nakonec navrací nejbližší shodu (Nejbližší je navrácen).

FLANN je zkratkou pro Fast Library for Approximate Nearest Neighbors, jedná se o C++ knihovnu. Ta obsahuje algoritmy optimalizované pro rychlé hledání nejbližšího souseda ve velkém objemu dat a pro významné body o velkých dimenzích. FLANN porovnávač by měl být rychlejší než Brute-Force v případě velkých objemů dat.

Oba principy mají svoji vlastní metodu. Dále následuje popis nejdůležitějších bodů, tj. jakým způsobem byly tyto metody implementovány.

Vlastní metody

Aby mohl algoritmus vyhledat nové místo výskytu objektu v modelu, potřebujeme snímek aktuální scény, obrázek hledaného objektu a vybrat metodu pro nalezení a popis významných bodů v obraze.

Obrázek hledaného objektu získáme obdobně jako v případě OpenCv trackerů pomocí inicializační metody initialize. Zde opět poskytneme jako vstup snímek scény s objektem. Po zavolání metody se objeví okno s naší scénou a na uživateli je vybrat oblast s předmětem tak, aby tam byl pokud možno celý a zároveň bylo ve vybrané oblasti

co nejméně pozadí. Výběr se provádí pomocí kurzoru myši, kdy kliknutím a tažením vybereme obdélníkovou oblast. Výběr je potvrzen stiskem klávesy „space“ nebo „enter“. Metoda poté vrátí souřadnice a rozměry vybrané oblasti, a tato oblast je ze snímku vybrána jako reference pro hledání. Takto vrácený obdélník je poté použit pro extrakci dané části scény a je uložen jako obraz modelu.

Pokud tedy máme obraz modelu, můžeme přistoupit k detekci objektu v dalších snímcích scény. Samotné metody jsou si velmi podobné. Metoda pro metodu Brute-force se nazývá `MatchImageBrurteForce` a pro FLANN se nazývá `MatchImageFLANN`. Tady je možné vidět jejich předpis:

```
private static PointF[] MatchImageBrurteForce(Image<Gray, byte> templateImage, Image<Gray, byte> sceneImage, FeatureDetectorMetod method){
}
private static PointF[] MatchImageFLANN(Image<Gray, byte> templateImage, Image<Gray, byte> sceneImage, FeatureDetectorMetod method)
{
}
```

Obě metody vyžadují stejné vstupní parametry, kterými jsou obrázek hledaného objektu ve stupních šedi `templateImage,` aktuální snímek ve stupních šedi `sceneImage` a metoda detekování a popisu významných bodů, která se má použít, parametr `method`. Jedná se o proměnou vlastního výčtového typu `FeatureDetectorMetod.`

Na začátku obou metod dochází k vytvoření a inicializaci proměnných, které jsou pak dále používány. Poté je vytvořen detektor významných bodů, jako třída `Feature2D`, na základě předaného parametru `method`. Dále je použita metoda detektoru pro výpočet a popis významných bodů, které je předán obrázek a ta vypočítá polohu významných bodů a jejich deskriptory navrátí v podobě `VectorOfKeyPoint` a `Mat`. Tato metoda je použita pro výpočet a popis, jak pro snímek scény, tak pro obraz sledovaného objektu. Poté je vytvořena instance třídy feature matcher, které jsou předány deskriptory významných bodů hledaného objektu a aktuálního snímku získané pomocí detektoru a použité metody pro jejich výpočet. Na základě těchto deskriptorů a nastavení matcheru jsou za pomoci metody `KnnMatch` určeny nejlepší shody mezi významnými body objektu a aktuálního snímku. Počet nejlepších shod pro každý významný bod objektu, který je vrácen jako `VectorOfVectorOfDMatch matches,` je dán parametrem „k“ metody `KnnMatch`. V našem případě jsme volili hodnotu parametru `k=2`, takže jsme pro každý bod dostali dvě nejlepší shody. Jelikož však chceme uvažovat jen shody, které nám jednoznačně určují novou pozici hledaného objektu, vyfiltrujeme si tyto shody pomocí metody `VoteForUniqueness`, té předáme naše shody spolu s parametrem `UniquenessThreshold`—jehož hodnota může dosahovat hodnoty od 0 do 1. Hodnota značí-poměr mezi matematickými vzdálenostmi oněch nejlepších dvou shod, pro který je shoda považována za unikátní. Shody (také si můžeme představit jako významné body) jejichž poměr mezi dvěma nejlepšími shodami je tedy menší než zadaný treshold jsou

odfiltrovány. Doporučená hodnota se pohybuje okolo 0,8. Metoda nám vrací masku, kde unikátní nejlepší shody jsou detekovány jako 255 a zbytek jako 0.

Pokud do tohoto kroku byl nalezen alespoň minimální počet shod splňující naše kritérium (obecně pro detekci je minimum 4 shody, ale problémem je potom spolehlivost), pokračujeme vyfiltrováním shod za pomocí metody `VoteForSizeAndOrientation`. Ta vyfiltruje na základě pozic významných bodů a jejich shod pouze ty shody, které vyhovují kritériu, že mohlo dojít ke zvětšení nebo zmenšení objektu v maximální míře, jaká je uvedena v parametru `scaleIncrement`, a že došlo k natočení pouze pro počet úhlů daný parametrem `rotationBins`. Každý interval povolených natočení je pak široký n*360°/`rotationBins` v úhlech. Metoda vrací upravenou masku shod.

Pokud i po předchozím kroku máme dostatek shod, použijeme metodu `GetHomographyMatrixFromMatchedFeatures`, abychom na základě klíčových bodů, shod a masky získali matici homografické transformace. Tato metoda pracuje na základě algoritmu RANSAC. Matici transformace využijeme pro transformaci perspektivy obdélníku tak, abychom dostali čtyři body, které znázorňují čtyřúhelník, ve kterém se nachází sledovaný objekt. Tyto body jsou pak navráceny jako návratová hodnota metody.

Co se týče parametrů feature matcheru třídy `BFmatcher`, tak jediný parametr, který jsme definovali, je `distanceType`, tj. podle jakého typu vzdálenosti se mají významné body posuzovat. Bylo dáno na doporučení a byly použity na základě použitého detektoru dva typy výpočtu vzdálenosti a to `DistanceType.L2` (Eukleidovská metrika) nebo `DistanceType.Hamming` (hamingova vzdálenost).

Co se týče metody využívající FLANN, postup algoritmu je stejný jako u Brute-Force matchingu. Jediným rozdílem je použití jiné třídy pro hledání shod. Jedná se o třídu `FlannBasedMatcher`, u které jsme zvolili pro parametrizaci metodu `KDTreeIndexParams`, která udává, že vytvořený index bude sestávat ze sady randomizovaných kd-trees, které budou prohledávány paralelně. Počet paralelních kd-trees je dán parametrem trees. Větší počet kd-trees dává lepší přesnost, ale zpomaluje algoritmus.

Když proběhne metoda úspěšně, tak vrátí vektor bodů, který představuje čtyřúhelník, ve kterém se na daném snímku sledovaný objekt nachází.

Nastavování parametrů algoritmů pro vyhledávání a popis významných bodů

Jednotlivé algoritmy, které můžeme využít, mají své vlastní nastavení parametrů v závislosti na tom, o jaký algoritmus se jedná. Byla testována různá nastavení těchto algoritmů a v rámci toho se dospělo k následujícím závěrům.

Defaultní doporučené parametry dávají poměrně robustní výsledky, proto je ve většině případů není potřeba měnit. Pokusili jsme se zpřísnit u jednotlivých algoritmů parametry podmínek pro nalezení významného bodu, jako jsou tresholdy apod., abychom zmenšili počet shod a tím zrychlili celý algoritmus. Výsledkem bylo, že zrychlení bylo

zanedbatelné a na druhou stranu se zvýšil počet případů, kdy jsme nebyli schopni objekt v obraze detekovat. Proto zpřísňování těchto parametrů není vhodné.

Poté jsme zkusili i druhý přístup a parametry pro detekci významných bodů jsme snížili. To samozřejmě vedlo k detekci většího počtu významných bodů. Tato skutečnost v některých případech napomáhala při detekci objektů, které se natáčely v osách jiných než Z kamery. Zlepšení však nebylo tak velké, aby se vyplatilo jej použít na úkor vyšší výpočetní náročnosti. Byl ponechán pouze jeden příklad pozměněných parametrů, a to úprav parametrů detektoru SIFT, kdy contrastThreshold byl nastaven na hodnotu 0,03 a edgeThreshold nastaven na hodnotu 11.

Abychom mohli efektivně měnit a nastavovat tyto parametry, bylo by potřeba znát přesnou situaci detekce a detekovaného objektu předem. Proto je lepší ponechat defaultní hodnoty, jelikož ty poskytují obstojné výsledky pro poměrně širokou škálu užití a situací.

2.3.2 Implementace – hardwarová akcelerace EmguCV Cuda

Metoda Feature matching je jedinou metodou, která disponuje i verzí využívající možnosti hardwarové akcelerace za pomoci jednotky GPU. Hardwarová a softwarová architektura Cuda funguje pouze na čipech od výrobce nVIDIA Corp. To použití této modifikace omezuje pouze na počítače vybavené grafickými čipy od tohoto výrobce.

Z dvojice metod Brute-Force a FLANN je metoda matchingu Brute-Force jediná s implementací pro architekturu Cuda, proto je také jako jediná takto implementována. Podobná situace je i na poli detektorů a deskriptorů významných bodů, kde ve verzi knihovny EmguCV 4.4 se nachází pouze detektor ORB (Oriented FAST and Rotated BRIEF), který se dá využít ve spojení s Brute-Force matchingem, který má taktéž Cuda implementaci. Cuda verze Brute-Force matchingu byla implementována v rámci stejné metody společně s klasickou Brute-Force metoda využívající CPU.

V rámci implementace algoritmu ORB, jak je implementován v Cuda, je schována vlastnost, která není nikde v dokumentaci popsána a může způsobovat problémy. ORB zpracovávaný na GPU vrací jen omezený počet detekovaných významných bodů stejně jako v případě CPU implementace, tento počet je metodě předáván jako parametr. To samo o sobě není problém, pokud by však algoritmus nevracel významné body na principu „kdo dřív příjde“. Pak může nastat situace, kdy se nám vrátí významné body detekované pouze ve vrchní čtvrtině snímku, jelikož tou dobou, když algoritmus prošel první třetinu snímku, zaplnil požadovaný počet hledaných významných bodů, a proto již dál nehledal. To je fundamentální rozdíl vzhledem k CPU implementaci, ta vrací detekované významné body na bázi nejlepších detekovaných. Aby tedy nedocházelo k situacím, kdy by významné body byly nalezeny jen v části snímku, bylo nutné předimenzovat maximální množství hledaných významných bodů. Celý algoritmus kvůli této skutečnosti byl mírně zpomalen, ale i tak byl zhruba dvakrát až třikrát rychlejší jak

ORB Brute-Force matching zpracovávaný na CPU, přičemž vracel zhruba pětkrát více významných bodů.

V okamžiku, kdy došlo k samotnému kroku implementace tohoto algoritmu počítaného z větší části na GPU, nastal problém v podobě RGB kamery, kterou je manipulátor vybaven. Cuda implementace funguje pouze v 64 bitových aplikacích a kamera primárně funguje 32 bitově. Drivery pro kameru se mi nakonec nepodařilo zprovoznit pro 64-bitovou implementaci, proto nebylo možné otestovat jaký a jak velký vliv by mělo toto urychlení algoritmů. Na základě pozorování při testování mimo manipulátor lze teoreticky předpokládat, že by mohlo dojít díky kratší výpočetní době ke zrychlení cyklu sledování a tím i k větší plynulosti. Rovněž díky několikanásobně většímu počtu vyhledaných významných bodů mohlo být zpřísněno kritérium shody při zachování dostatečného počtu shod, což vede ke zvětšení potenciálu v rámci přesnosti a robustnosti. Velký počet významných bodů a shod také napomáhá zmenšovat vliv snímání objektu z jiných úhlů, než byl úhel pořízení modelového snímku pro hledání předmětu v obraze.

Obrázek 2.2 Ukázka principu Feature Matching(horní před filtrováním, spodní obrázek po odfiltrování nevhodných shod)

2.3.3 Kdy může metoda selhávat

Vzhledem ke skutečnosti, že algoritmus prohledává celý snímek a hledá významné body, mezi kterými pak hledá shodu s významnými body hledaného objektu, může dojít k situaci, kdy se ve snímku vyskytne nějaký jiný objekt, který je v danou chvíli víc podobný sledovanému objektu, má dle použitého kritéria lepší shodu a z principu hledání nejlepší shody je označen jako hledaný objekt. Tento jev se může objevovat převážně tehdy, pokud se ve snímané scéně vyskytuje více stejných objektů jako je objekt, který

sledujeme nebo je objekt součástí opakující se struktury (například okna, kus střechy, trávník apod). Tento problém se může částečně eliminovat tím, že pomocí přidaného kódu se přiblížíme k trackovacím algoritmům tak, že omezíme oblast hledání na oblast kolem posledního výskytu objektu. Tento problém se může objevit nezávisle na tom, zda použijeme metodu porovnávání Brute-Force nebo FLANN. Tento způsob je však spíše vhodný pro statické snímání za pomocí kamery.

Vzhledem k povaze a principu metody, kdy jde o hledání nejlepší shody, je velice obtížné stanovit podmínku pro absenci sledovaného předmětu v obraze, neboť má-li sledovaná scéna a objekt velké množství detekovaných významných bodů, je velká pravděpodobnost, že se mezi významnými body sledované scény a sledovaného objektu vyskytnou shody splňující námi stanovená kritéria shody a automaticky jsou tedy na základě principu nejlepší shody označeny jako body sledovaného objektu, přestože se sledovaný objekt vůbec nemusí ve snímané scéně nacházet.

2.3.4 Souhrn metod feature matching

Souhrn

Obě metody dosahují velmi srovnatelných výkonů. To je možná také zapříčiněno dodatečnými filtracemi shod, které byly prováděny poté, co byly shody vygenerovány. Mohou být sice pomalejší, při využití některých algoritmů pro nalezení významných bodů (AKAZE aj.), nežli trackovací algoritmy, ale všeobecně mají velkou spolehlivost a přesnost, tedy za předpokladu, že se sledovaný objekt ve scéně nachází. V opačném případě mohou detekovat velké množství falešných detekcí. To lze omezit zvýšením podmínky pro minimální počet profiltrovaných shod, což ovšem na druhou stranu může mít za následek v některých situacích neschopnost detekce objektu v obrazu. Metoda FLANN je o něco rychlejší nežli Brute-force, ale v rámci naší implementace zpravidla nešlo o velké rozdíly. Výhodou metod feature matching z hlediska návrhu je, že nad jejich implementací máme větší kontrolu a lze je tak lépe ladit. Bohužel nelze nalézt ideální nastavení parametrů pro algoritmy vyhledávání významných bodů a matchery tak, abychom zajistili vysokou spolehlivost ve většině případů pro většinu scén a kamer. Ideální by bylo mít individuální nastavení parametrů pro jednotlivé systémy a scény, popřípadě i jednotlivé typy sledovaných objektů.

Návrhy na zlepšení

Jedná se o algoritmus, který není dokonalý a má možnosti se zlepšovat. Jako jedna z možností bylo zamýšleno ho více přiblížit algoritmům sledovacím. Avšak předpovídat polohu na bázi nějakých minulých výskytů v obrazu se ukázalo v případě OpenCV sledovacích algoritmů jako spíše přitěžující, kdy v případě naší implementace pro pohybující se kameru na manipulátoru chceme udržet objekt na jenom místě záběru. Proto místo pro zlepšení leží spíše v aktualizování modelového snímku pro porovnávání. To by mohlo vést ke zlepšení robustnosti algoritmu proti změně světelných podmínek a natáčení

objektu mimo rovinu obrazu. Hlavním problémem by však bylo stanovení podmínek, které by určovali, v jakém okamžiku (při jakých podmínkách) takovouto aktualizaci provést, aby to mělo očekávaný účinek a došlo ke zlepšení algoritmu.

Dalším zlepšením by mohl být vznik algoritmu na podobném principu, také využívající feature matching, který by však nevyužíval jako model snímek objektu, ale jeho 3D model významných bodů. Tyto body by byly následně porovnávány s detekovanými body v obraze a shody by byly mapovány tak, že by došlo k odhadu polohy objektu v 3D nebo až 6D prostoru. Jednalo by se však o poměrně komplexní a náročnější úlohu, ať už jde o získávání informací pro model nebo samotnou estimaci.

2.4 Metody sledování na bázi barvy v obraze

2.4.1 HSV

Jedná se o barevný model, může být také označován jako HSB. Představuje barevný prostor podobný známějšímu BRG barevnému modelu, lze ho také považovat za jeho alternativní reprezentaci. HSV lépe reflektuje vnímání barev lidským okem. Také na něj můžeme nahlížet tak, že modeluje vzhled barev v závislosti na působení světla. Dokážeme si tedy intuitivně představit, jaká bude výsledná barva, pokud některou hodnotu složek změníme. Geometrickou reprezentací HSV modelu je válec. Jednotlivé barvy v rámci modelu jsou reprezentovány trojicí hodnot, stejně jako u RGB modelu. HSV je tedy zkratkou názvů jeho tří složek hue, saturation a value. Hue představuje odstín barvy, jde vlastně o samotnou barvu, které lidé přidělují názvy. Její hodnota je reprezentována pomocí stupňů na standartním barevném kole, tedy od 0° do 360°. Saturation reprezentuje množství šedi v odstínu. Může být také označována jako sytost. Vzrůstá směrem ke kraji válce. Její hodnota stoupá od nesaturované, přes stupně šedi po plně saturovanou. Poslední složkou je value, která vyjadřuje jas nebo také intenzitu barvy. Můžeme ji rovněž vnímat jako množství přidávané černé do základní barvy. HSV model se často využívá v úlohách zpracování obrazu, kdy je potřeba objekty segmentovat v závislosti na jejich barvě.

2.4.2 Histogram

Všeobecně je histogram pojem ze statistiky, kdy se jedná o grafické znázornění distribuce určitých dat. Využívá se sloupcových grafů, které jsou rozděleny v celém spektru na sloupce o stejné šířce. Sloupec tedy udává šířku intervalu a výška sloupců označuje četnost sledované veličiny v tomto intervalu.

V počítačovém vidění a obecně u fotografií je histogram používán pro zobrazení rozložení nějaké vlastnosti obrazu. Výška jednotlivých sloupců v tomto případě značí počet pixelů, které mají v obraze danou hodnotu. Počet sloupců typicky nabývá rozsahu hodnot, které můžou jednotlivé pixely nabývat, existuje pak sloupec pro každou z možných hodnot. Sledovanou veličinou obrazu bývá intenzita jednotlivých základních

složek RBG spektra, nebo HSV složek. Histogram však nemusí být sestrojován pouze na základě informací o barvě, ale mohou to být data o jakékoliv vlastnosti, která je užitečná pro popis obrazu a dokážeme ji měřit (gradient, směr apod.).

2.4.3 Metoda sledování markerů

Jedním z bodů zadání bylo navrhnout sledovací algoritmus na bázi sledování markerů. Jedná se o běžný a často využívaný způsob sledování, jelikož u něj odpadá problém nutné všestrannosti, který sebou přináší rozdílné chování i výkonnost v různých situacích, prostředích a podmínkách.

Jako markery byly pro sledování zvoleny tři barevné kruhy, jejichž barva je k jejich okolí kontrastní. Tyto markery musí ve snímku společně tvořit různostranný trojúhelník, nesmí se tedy jednat ani o rovnostranný, ani o rovnoramenný útvar. Volba tří bodů je opodstatněna z hlediska zpětné kontroly, tj. zda se hledaný obrazec markerů v dané scéně nachází. Je totiž možné najít jeden nebo dva útvary, které by mohly splnit definované podmínky pro naše markery, ale je nepravděpodobné, že by se ve scéně objevily tři objekty, které by splňovaly podmínky pro dané markery. Zároveň jsme schopni díky tomu, že nesledujeme jeden ale více bodů, získat natočení v ose Z kamery. Jelikož je v rámci tohoto zadání k dispozici pouze jedna kamera a nemáme pevně dané rozměry markerového trojúhelníku, přicházíme o informaci ohledně natočení ve zbylých dvou osách X a Y. Existují i metody mapování předmětů pomocí jedné kamery do trojrozměrného souřadného systému, ty v rámci zadané úlohy nelze využít, ať už je to kvůli nutnosti snímat objekt z více různých úhlů, nebo potřebou předem disponovat přesnými informacemi o rozměrech sledovaného objektu.

Detekce markerů v obraze probíhá na základě prahování obrazu a následném vyhledávání kontur. V tomto případě kontura znamená křivku spojující všechny spojité body podél hranice. Jedná se tedy o obrys využívaný jako nástroj pro analýzu, detekci a rozpoznávání tvarů a objektů v počítačovém vidění.

Prvním krokem při hledání kontur je prahování. To děláme, pokud hledáme kontury na základě jejich jedinečné barvy, která je odlišuje od okolí. Pokud by byl naopak primárním znakem našich hledaných kontur tvar, je v tomto případě vhodnější zvolit namísto prahování některý z hranových detektorů, například canny detektor. Tento přístup lze zvolit, jelikož výstupem obou postupů je binární obraz, ve kterém jsou poté hledány kontury. V našem případě jsme si mohli vybrat, zvolili jsme jednodušší variantu, tedy prahování. Prahování je prováděno na snímcích s barevnou informací v HSV formátu. Tento formát je využíván kvůli jeho spojitosti barev v závislosti na osvětlení, tedy takzvaných odstínů. Ze vstupního obrazu se na základě spodní a horní barevné meze vytvoří binární obraz, ve kterém se hledají shluky, ze kterých se poté počítají kontury. Kontury jsou následně rozlišovány mezi sebou na základě specifických geometrických vlastností vyplývajících z jejich tvaru.

Všeobecně se jedná o principiálně nejjednodušší použitou metodu sledování. Toto zjednodušení si můžeme dovolit díky striktnímu definování podoby sledovaného objektu, což jsme si u ostatních metod dovolit nemohli. S tímto zjednodušením jde také ruku v ruce úspora výpočetní náročnosti, k výpočtu jsou využívány v zásadě jen základní maticové výpočty a operace.

Implentace

Samotná struktura metody spočívá ve vyhledávání kontur na základě prahování HSV snímku. Vstupem algoritmu je tedy HSV obraz, ve kterém chceme najít hledané kontury a limity HSV barev pro prahování.

Aby nemusely být limitní hodnoty barev odhadovány, popřípadě nastavovány jiným experimentálním způsobem, jako jsou třeba posuvníky, byla v rámci implementace vytvořena i metoda pro získání těchto limit přímo z obrazu.

Tato metoda přijímá jediný parametr a tím je HSV obraz, na kterém se nachází naše markery, ze kterých chceme získat limity pro prahování. Po zavolání této metody je uživateli zobrazeno okno s obrazem a na uživateli je, aby vybral kliknutím a tažením myši, kdy se objeví vybraný obdélník, nejlépe vnitřek některého z markerů.

S takto vybraným kusem snímku se vypočte histogram pro každou ze složek hue, saturation i value. Získané histogramy jednotlivých složek jsou zpracovány a z nich získána intenzita s největší četností. Tato hodnota je potom použita jako střed pásma barev, které chceme detekovat. Horní a spodní okraj tohoto pásma se pak stává spodní a horní limitní hodnotou pro prahování. Šířka pásma, která je používána, byla stanovena experimentálně tak, aby byl potlačen případný šum od ostatních barev. Skutečnost, že můžeme takto určit spojité pásmo pro prahování, je díky vlastnosti HSV modelu, který se mění spojitě.

Pokud bychom vycházeli jen z teorie, stačilo by nám pouze určit hodnoty limit pro složku hue a zbylé dvě složky brát v celém rozsahu. To ovšem ani při velkém kontrastu markerů s okolím nedopadá vždy úspěšně, proto je prováděna analýza pomocí histogramu všech tří složek. V EmguCV má složka hue modelu HSV rozsah pouze od 0 do 180 a červená barva je definována okolo 0 a zároveň i okolo 180. To má za následek, že automatický výběr limit pro červenou barvu nemusí vždy fungovat, proto je vhodné zvolit jinou barvu než červenou a jí blízké barvy. Navíc v některých případech, kdy limity hue pro prahování nabývají hodnoty 0 nebo 180, dochází k neočekávaným výsledkům prahování, proto byl omezen rozsah pro limity v rámci složky hue na 1-179.

Když tedy máme HSV limity pro prahování, provedeme prahování na aktuálním snímku (funkce InRange). Jako výstup získáme šedo-tónový obraz, také bychom ho mohli označit za binární, kdy pixely, které měly barvu v intervalu zadaných limit, mají hodnotu 255 a ostatní mají hodnotu 0.

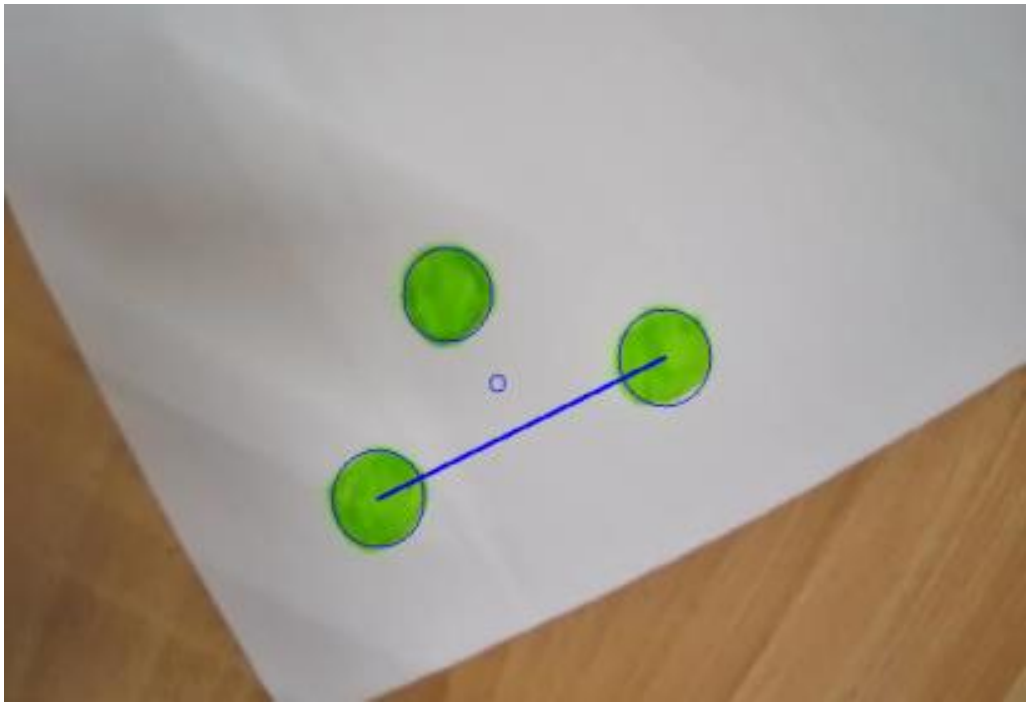
Na takto připravený binární obraz použijeme funkci pro vyhledávání kontur. Ta nám navrátí seznam nalezených kontur. Díky nastavení návratového typu kontur jako external, obdržíme jen „vnější“ kontury. To znamená, že pokud nějaká kontura obsahuje uvnitř další nalezené kontury, a tak je tedy obklopuje, je jako nalezená označena pouze ta největší a ty, které jsou jí obklopovány, jsou potlačeny. Toto nastavení je použito, protože se jím zmenšuje počet vrácených nálezů kontur a vyhovuje našim požadavkům, které klademe pro výběr markerů.

Nalezené kontury jsou poté filtrovány podle několika kritérií. Podle solidity, což je poměr mezi plochou kontury a plochou Hullovy oblasti. Hullova oblast je oblast obepínající konturu takovým způsobem, že všechny křivky, které tvoří její obvod, jsou konvexní, tedy „vyboulené“ ven nebo alespoň rovné. Dalším sledovaným kritériem je minimální šířka. Posledním kritériem je poměr mezi poloměrem kruhu, který konturu co nejlépe obklopuje (kruh s nejmenším obsahem) a ekvivalentním poloměrem, který by měl kruh o stejném obsahu, jako má plocha kontury. Na základě těchto kritérií jsou pak vybírány kontury blízké kruhu.

Takto profiltrované kontury jsou seřazeny podle velikosti plochy a za hledané kontury jsou označeny tři největší. Jako střed tělesa je označen geometrický střed trojúhelníku, jehož vrcholy leží ve středech kružnic nejlépe obepínajících výsledné kontury.

Kvůli striktním pravidlům pro výběr kruhu podobné kontury docházelo, při natáčení markerů v jiné ose než je osa Z kamery (snímání z úhlu), k nenalezení kontur, z důvodu nepodobnosti kruhu. A pokud nejsou nalezeny tři markery, tak je prohlášeno, že se hledaný objekt ve scéně nenachází. Proto byla implementována navíc zjednodušená metoda, kdy kontury nejsou filtrovány a jako shody se označí vždy pouze tři kontury s největší plochou.

U této metody je jako u jediné také implementováno sledování natáčení. To je realizováno porovnáváním vzájemné polohy nejdelší strany trojúhelníku, tvořeného markery, v různých snímcích. Proto je důležité, aby nebyly markery umísťovány jako rovnoramenné nebo rovnostranné trojúhelníky.



Obrázek 2.3 Ukázka detekce markerů s vyznačením středu a nejdelší strany

Funkčnosti a shrnutí

Díky tomu, že v tomto případě víme přesně, co sledujeme a jak to má vypadat, je algoritmus jednodušší, což sebou nese i snížení výpočetní náročnosti, a tedy i zrychlení. Lze pozorovat i velmi malé (v řádech ms) zrychlení i při porovnání verze s kruhovým kritériem a bez něj.

Metoda s kritériem hledání kruhu podobných markerů nám poskytuje poměrně přesnou lokalizaci markerů, a především spolehlivou identifikaci absence markerů v obraze. Na druhou stranu toto kritérium může selhávat v identifikaci markerů pokud neleží v rovině kolmé na osu Z kamery, popřípadě se nachází ve větší vzdálenosti od středu obrazu. Tato vzdálenost, kdy algoritmus začne selhávat, záleží mj. na tom, jak přesně udělané jsou markery a v jaké vzdálenosti jsou od kamery.

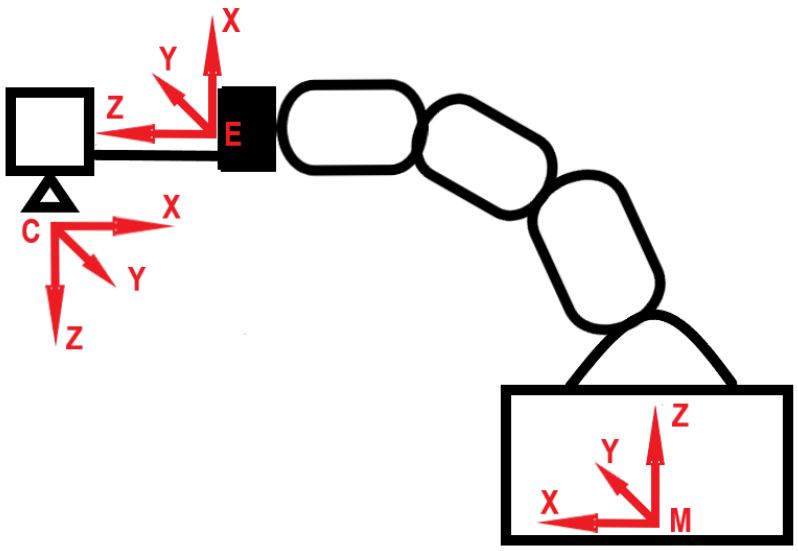
Kvůli těmto problémům byla implementována zjednodušená verze algoritmu, která hledá pouze tři největší markery v obraze. Ta je dostatečně spolehlivá pouze v případě, že jsou markery dost kontrastní k okolí a vždy jsou v obraze všechny tři. Pokud by se nějaký z markerů v obraze nenacházel, je pravděpodobné, že dojde ke špatné detekci.

Vhodnou barvou pro markery je taková, která je dostatečně kontrastní s okolím. Tmavé barvy nemívají nejlepší výsledky, proto je vhodné volit barvy světlejší. Mělo by se mj. jednat o barvu, u které v HSV modelu nejsou v sousedství barvy nacházející se v obraze, a která se nenachází v počátku HSV modelu, tedy červená barva (hue 0°/360°).

2.5 Určení nové pozice manipulátoru

Abychom mohli vykonávat úlohu sledování objektu za pomoci kamery a robotického manipulátoru, je zapotřebí určit novou polohu manipulátoru na základě obrazových dat z kamery a vazeb mezi jednotlivými členy řetězce.

Celý řetězec lze rozdělit na několik samostatných systémů, kdy každý bude mít svůj vlastní souřadný systém. Pro polohování manipulátoru využijeme možnost zadat požadovanou polohu koncového bodu jako polohu koncového bodu, tj. jako souřadnice v systému manipulátoru. Akční řetězec si tedy rozdělíme na tři souřadné systémy. Kdy budeme systém manipulátoru M brát jako základní systém pro výpočet nové polohy. Počátek tohoto systému se nachází na středu spodní strany platformy, která slouží k uchycení robotu. Dále si definujeme systém E (efector), jehož počátek se nachází v koncovém bodu manipulátoru. Jako poslední systém budeme uvažovat systém C (camera), jehož počátek se nachází na středu čočky kamery.



Obrázek 2.4 Schématické znázornění souřadných soustav manipulátoru

2.5.1 Určení polohy výskytu objektu ve snímku

Prvním krokem k tomu, abychom mohli objekt sledovat a vypočítat novou polohu manipulátoru, je určit polohu objektu v aktuálním snímku. Poloha objektu je zjišťována z obrazových dat, které zachycujeme za pomoci kamery. Po zpracování aktuálního snímku jednou z metod popsaných v předchozí kapitole dostaneme bod v souřadném systému kamery, který indikuje, že se v něm vyskytuje sledovaný objekt. Tento bod považujeme za střed sledovaného objektu a je určován v závislosti na použité metodě. Budeme se držet premisy, že chceme udržet sledovaný objekt ve středu obrazu. Proto algoritmy vrací souřadnice v pixelech představující aktuální pozici ve snímku, kdy jeho

souřadný systém má počátek v levém horním rohu snímku, souřadnice *x* roste směrem k pravému okraji a souřadnice *y* roste směrem ke spodnímu okraji snímku.

Tímto způsobem dostaneme pouze souřadnice v pixelech. Jelikož pro určení polohy využíváme informace pouze z jediné RGB kamery, pro převod souřadnic do mm potřebujeme znát vzdálenost kamery od měřeného objektu *d*. Tu lze získat za pomoci laserového skeneru, ten má však pouze malý omezený měřící rozsah. V ostatních případech jsme v rámci této aplikace s využitím dostupných prostředků odkázáni na to, že je nutno vzdálenost předmětu od kamery před začátkem sledování změřit a předpokládat, že se v časem nebude měnit. Poté budeme předpokládat, že celá rovina se nachází ve vzdálenosti *d* ve směru osy Z od počátku souřadného systému kamery. Dále potřebujeme znát zorný úhel kamery vertikální α_v a horizontální β_h. Tyto úhly jsem převzal z parametrů projektu RoScan, jelikož v dokumentaci, kterou jsem k použité kameře nalezl, tyto informace uvedeny nebyly.

Pokud známe výše zmíněné hodnoty, můžeme převést polohu sledovaného objektu v rámci snímku z pixelů na mm za pomocí následujících dvou rovnic

$$x_{mm} = x_{px} \cdot \frac{d \cdot tan\left(\frac{\beta_h}{2}\right)}{\frac{R_h}{2}}, \tag{2.1}$$

$$y_{mm} = y_{px} \cdot \frac{d \cdot tan\left(\frac{\alpha_v}{2}\right)}{\frac{R_v}{2}}, \tag{2.2}$$

kde x_{mm} a y_{mm} jsou souřadnice středu objektu v mm, x_{px} a y_{px} jsou souřadnice středu objektu v pixelech, R_v a R_h je vertikální a horizontální rozlišení kamery v pixelech (počet pixelů), *d* je vzdálenost předmětu/snímané scény od kamery, α_v a α_h jsou vertikální a horizontální zorné úhly kamery. Tímto způsobem tedy lze získat souřadnice objektu v souřadném systému roviny snímané plochy.

Problém nastává v okamžiku, když nejsme schopni měřit vzdálenost objektu a objekt se vzdaluje, či přibližuje ke kameře. V takovém případě dochází ke zkreslování výsledků z důvodu ztráty informace o vzdálenosti.

Pokud bychom znali přesné rozměry sledovaného obrazce a dokázali bychom jej dobře detekovat na základě znalosti velikosti takového objektu v obraze a jeho skutečných rozměrů, byli bychom schopni vypočítat, jakou vzdálenost představuje jeden pixel ve skutečnosti. Tímto by se dalo částečně kompenzovat zkreslení způsobené přibližováním a vzdalováním sledovaného objektu od kamery. V případě, že známe jakou délku představuje jeden pixel, můžeme na základě této informace spolu s úhly kamery odhadnout vzdálenost objektu od kamery. Tyto výpočty by však počítaly s premisou toho, že objekt leží v předpokládané rovině snímku, tedy v rovině kolmé na osu *Z* kamery. Tudíž tato metoda přestane fungovat, pokud by docházelo k tomu, že by byl uvažovaný

objekt použitý pro přepočet natočen kolem osy X nebo Y a nebyl by tak kolmý k ose souřadnicového systému kamery. $H_{MCnew} = H_{ME}H_{EC}T_{xyz}R_z$

2.5.2 Homogenní transformace

Abychom mohli zjistit souřadnice nové polohy kamery na základě aktuální pozice středu sledovaného objektu v souřadnicovém systému manipulátoru M, využijeme vlastností homogenní transformace. Pro určení požadovaných souřadnic (x, y, z, roll, pitch yaw) pro novou polohu kamery musíme skloubit dohromady několik homogenních transformací mezi sousedními souřadnými systémy, které jsme si nadefinovali a známe vazby mezi nimi. Homogenní transformace nové pozice kamery do souřadnicového systému manipulátoru se tedy bude skládat z několika dílčích homogenních transformací. Nejprve je zapotřebí určit homogenní transformaci z kamery do manipulátoru v okamžiku zachycení snímku z transformace manipulátor-efektor a efektor-kamera

$$H_{MC} = H_{ME}H_{EC}. \tag{2.3}$$

Homogenní souřadnice bodu v souřadnicovém systému kamery vyjádřené jako souřadnice v souřadnicovém systému manipulátoru, dostaneme pomocí vztahu

$$P_M = H_{MC}P_C\ . \tag{2.4}$$

Na základě homogenní transformace manipulátor-kamera plus translace a rotace založené na nové poloze objektu ve snímku zjistíme homogenní transformaci mezi manipulátorem a novou požadovanou polohou systému kamery. Vztah pro výpočet této transformace je následující

$$H_{MCnew} = H_{MC}H_{CCnew}\ . \tag{2.5}$$

Jelikož manipulátor pozicujeme pomocí cílové pozice efektoru a ne pomocí pozice kamery, tak musíme dopočítat transformaci pro manipulátor-efektor_nová (nová pozice efektoru) na základě transformace manipulátor-kamera_nová a transformace kamera-efektor. Vyplyne nám tedy vztah

$$H_{MEnew} = H_{MCnew}H_{CE}\ . \tag{2.6}$$

Níže jsou blíže popsány jednotlivé homogenní transformace mezi sousedními systémy.

Transformace z nové polohy do souřadnicového systému kamery

Máme bod P, který je dán souřadnicemi středu nové polohy sledovaného objektu. Jeho homogenní souřadnice jsou

$$P = \begin{bmatrix} x_C \\ y_C \\ d \\ 1 \end{bmatrix}^T , \tag{2.7}$$

kde x_C a y_C jsou souřadnice těžiště v souřadnicovém systému kamery a d je vzdálenost předmětu od kamery. Tento bod později použijeme jako základ pro výpočet transformace mezi starou a novou polohou kamery.

Transformace ze souřadnicového systému kamery do souřadnicového systému koncového bodu

Kamera se nachází na držáku, který je připevněn na koncovém bodu manipulátoru, tedy na efektoru spolu s termální kamerou a laserovým skenerem. Střed koncového bodu nám pak určuje počátek souřadnicového systému efektoru E. Homogenní transformace mezi systémy tedy bude kombinací translace a rotace kolem všech os dle modelu[12][13]

$$H_{EC} = Trans(x_t, y_t, z_t)\,Rot_z(\alpha)\,Rot_y(\beta)\,Rot_x(\gamma) =$$
$$\begin{pmatrix} 1 & 0 & 0 & x_t \\ 0 & 1 & 0 & y_t \\ 0 & 0 & 1 & z_t \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c(\alpha) & -s(\alpha) & 0 & 0 \\ s(\alpha) & c(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c(\beta) & 0 & s(\beta) & 0 \\ 0 & 1 & 0 & 0 \\ -s(\beta) & 0 & c(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & c(\gamma) & -s(\gamma) & 0 \\ 0 & s(\gamma) & c(\gamma) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} =$$
$$= \begin{bmatrix} c(\alpha)\cdot c(\beta) & c(\alpha)\cdot s(\beta)\cdot s(\gamma) - s(\alpha)\cdot c(\gamma) & c(\alpha)\cdot s(\beta)\cdot c(\gamma) + s(\alpha)\cdot s(\gamma) & x_t \\ s(\alpha)\cdot c(\beta) & s(\alpha)\cdot s(\beta)\cdot s(\gamma) + c(\alpha)\cdot c(\gamma) & s(\alpha)\cdot s(\beta)\cdot c(\gamma) - c(\alpha)\cdot s(\gamma) & y_t \\ -s(\beta) & c(\beta)\cdot s(\gamma) & c(\beta)\cdot c(\gamma) & z_t \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.8}$$

kde c(x) = cos(x) a s(x) = sin(x), x_t, y_t a z_t jsou pak posunutí systému koncového dobu do systému kamery podél příslušných os, α je natočení kolem osy z, β je natočení kolem osy y a γ je natočení kolem osy x systému kamery C okolo os souřadnicového systému koncového bodu E. Jedná se o parametry, které jsou dány upevněním kamery ke koncovému bodu.

Transformace ze souřadnicového systému koncového bodu do souřadnicového systému manipulátoru

Aktuální pozice koncového bodu dostáváme z robota jako souřadnice v souřadnicovém systému manipulátoru M. Souřadnice obdržíme ve formátu roll, pitch, yaw, x, y, z. Proto je transformace ze systému koncového bodu E do systému manipulátoru M [12][13]

$$H_{ME} = Trans(x, y, z\,)Rot_z(\alpha)Rot_y(\beta)Rot_x(\gamma) =$$
$$\begin{pmatrix}1 & 0 & 0 & x\\ 0 & 1 & 0 & y\\ 0 & 0 & 1 & z\\ 0 & 0 & 0 & 1\end{pmatrix}\begin{pmatrix}c(\alpha) & -s(\alpha) & 0 & 0\\ s(\alpha) & c(\alpha) & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}\begin{pmatrix}c(\beta) & 0 & s(\beta) & 0\\ 0 & 1 & 0 & 0\\ -s(\beta) & 0 & c(\beta) & 0\\ 0 & 0 & 0 & 1\end{pmatrix}\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & c(\gamma) & -s(\gamma) & 0\\ 0 & s(\gamma) & c(\gamma) & 0\\ 0 & 0 & 0 & 1\end{pmatrix} =$$
$$= \begin{bmatrix} c(\alpha)\cdot c(\beta) & c(\alpha)\cdot s(\beta)\cdot s(\gamma) - s(\alpha)\cdot c(\gamma) & c(\alpha)\cdot s(\beta)\cdot c(\gamma) + s(\alpha)\cdot s(\gamma) & x\\ s(\alpha)\cdot c(\beta) & s(\alpha)\cdot s(\beta)\cdot s(\gamma) + c(\alpha)\cdot c(\gamma) & s(\alpha)\cdot s(\beta)\cdot c(\gamma) - c(\alpha)\cdot s(\gamma) & y\\ -s(\beta) & c(\beta)\cdot s(\gamma) & c(\beta)\cdot c(\gamma) & z\\ 0 & 0 & 0 & 1\end{bmatrix},$$

(2.9)

kde c(x) = cos(x) a s(x) = sin(x), x, y a z jsou aktuální souřadnice koncového dobu v souřadnicovém systému manipulátoru M, α je natočení kolem osy z (roll), β je natočení kolem osy y (pitch) a γ je natočení kolem osy x (yaw).

Transformace mezi souřadnicovým systémem kamery a nově požadovaným souřadnicovým systémem kamery

Aktuální pozice kamery je známa a potřebujeme vyjádřit vztah mezi touto pozicí a nově vypočtenou pozicí pro kameru. Jelikož známe

$$H_{CCnew} = Trans(\Delta x, \Delta y, \Delta z\,)Rot_z(\alpha) =$$
$$\begin{pmatrix}1 & 0 & 0 & x_C - c_x\\ 0 & 1 & 0 & y_C - c_y\\ 0 & 0 & 1 & d - h\\ 0 & 0 & 0 & 1\end{pmatrix}\begin{pmatrix}c(\alpha) & -s(\alpha) & 0 & 0\\ s(\alpha) & c(\alpha) & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix} = \begin{bmatrix}c(\alpha) & -s(\alpha) & 0 & x_C - c_x\\ s(\alpha) & c(\alpha) & 0 & y_C - c_y\\ 0 & 0 & 1 & d - h\\ 0 & 0 & 0 & 1\end{bmatrix},$$

(2.10)

kde c(x) = cos(x) a s(x) = sin(x), Δx, a Δy jsou rozdíly souřadnic bodu P a souřadnic středu obrazu C, Δz je rozdíl aktuální vzdálenosti kamery od předmětu d a požadované vzdálenosti h, α je úhel natočení sledovaného objektu kolem osy Z kamery oproti požadovanému natočení.

Tato transformace by se dala samozřejmě rozšířit o natočení objektu i v dalších osách, pak by matice měla stejný tvar jako matice 2.8 nebo 2.9, pouze by obsahovala patřičné jiné hodnoty. V našem případě však ztrácíme informace o natočení v ostatních osách kvůli datům pouze z jedné kamery. Kompenzace, pro natočení v jedné další ose pomocí laserového skeneru, by byla za určitých striktních podmínek (rovný povrch, pohyb pouze určitým směrem, dostatečná velikost předmětu apod.) možná, ale byla by značně složitá a nebyla by dostatečně spolehlivá. Proto nebyla tato možnost implementována.

Transformace mezi souřadnicovým systémem manipulátoru a nově požadovaným souřadnicovým systémem kamery

Jak již bylo zmíněno na začátku této kapitoly, cílem není zjistit transformaci mezi

soustavou manipulátoru a novou soustavou kamery, nýbrž transformaci mezi soustavou manipulátoru a novou soustavou efektoru. Abychom požadovaného výsledku dosáhli, musíme spočítat rovnici číslo 2.6, kdy výsledkem je obecná transformace v následujícím tvaru[20]

$$H_{MEnew} = H_{MCnew}H_{CE} = H_{MCnew}(H_{EC})^{-1} = \begin{bmatrix} c(\alpha)\cdot c(\beta) & c(\alpha)\cdot s(\beta)\cdot s(\gamma) - s(\alpha)\cdot c(\gamma) & c(\alpha)\cdot s(\beta)\cdot c(\gamma) + s(\alpha)\cdot s(\gamma) & x \\ s(\alpha)\cdot c(\beta) & s(\alpha)\cdot s(\beta)\cdot s(\gamma) + c(\alpha)\cdot c(\gamma) & s(\alpha)\cdot s(\beta)\cdot c(\gamma) - c(\alpha)\cdot s(\gamma) & y \\ -s(\beta) & c(\beta)\cdot s(\gamma) & c(\beta)\cdot c(\gamma) & z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.11}$$

kde c(x) = cos(x) a s(x) = sin(x), x, y a z jsou nové souřadnice efektoru E v souřadnicovém systému manipulátoru M, α je natočení kolem osy z (roll), β je natočení kolem osy y (pitch) a γ je natočení kolem osy x (yaw).

2.5.3 Nová pozice manipulátoru

Za pomocí popsaných homogenních transformací z předchozí kapitoly jsme schopni určit transformaci pro novou požadovanou polohu kamery. Dále známe vztah mezi polohou kamery a koncového bodu efektoru, která je pevně určena upevněním kamery. Jsme tedy schopni určit požadovanou polohu koncového bodu manipulátoru. Tu potřebujeme k tomu, abychom mohli posílat manipulátoru příkazy pro přesun. Není však možné posílat do manipulátoru transformační matice, proto z transformační matice H_{MEnew} musíme získat souřadnice nové polohy efektoru ve tvaru roll, pitch, yaw, x, y, z. Matice H_{MEnew} je popsána rovnicí 2.11. Kombinací různých prvků a následnou úpravou dostáváme vztahy pro výpočet souřadnic a úhlů v následujícím tvaru [22]

$$\alpha = tan^{-1}\left(\frac{M_{21}}{M_{11}}\right) = tan^{-1}\left(\frac{sin(\alpha)\cdot cos(\beta)}{cos(\alpha)\cdot cos(\beta)}\right) = tan^{-1}\left(\frac{sin(\alpha)}{cos(\alpha)}\right), \tag{2.12}$$

$$\beta = tan^{-1}\left(\frac{-M_{31}}{\sqrt{1-M_{31}^2}}\right) = tan^{-1}\left(\frac{sin(\beta)}{\sqrt{1-(-sin(\beta))^2}}\right) = tan^{-1}\left(\frac{sin(\beta)}{\sqrt{cos^2(\beta)}}\right) = tan^{-1}\left(\frac{sin(\beta)}{cos(\beta)}\right), \tag{2.13}$$

$$\gamma = tan^{-1}\left(\frac{M_{32}}{M_{33}}\right) = tan^{-1}\left(\frac{cos(\beta)\cdot sin(\gamma)}{cos(\beta)\cdot cos(\gamma)}\right) = tan^{-1}\left(\frac{sin(\gamma)}{cos(\gamma)}\right), \tag{2.14}$$

$$\mathrm{x} = M_{14} = x, y = M_{24} = y, z = M_{34} = z\ , \tag{2.15}$$

kde x, y a z jsou nové souřadnice efektoru E v souřadnicovém systému manipulátoru M, α je natočení kolem osy z (roll) systému M, β je natočení kolem nové osy y (pitch) a γ je natočení kolem nové osy x (yaw), M_{AB} značí prvek matice homogenní transfomace, kdy A je číslo řádku a B je číslo sloupce (číslováno od 1).

Pokud takto vypočteme souřadnice nové polohy efektoru, mohou nastat dvě situace:

1. Vše je v pořádku, pozice je dosažitelná a můžeme poslat požadavek na pohyb.
2. Pozice je nedosažitelná.

V druhém případě nastává problém, kdy musíme rozhodnout kam manipulátor poslat, aby bylo možné nadále sledovat náš objekt.

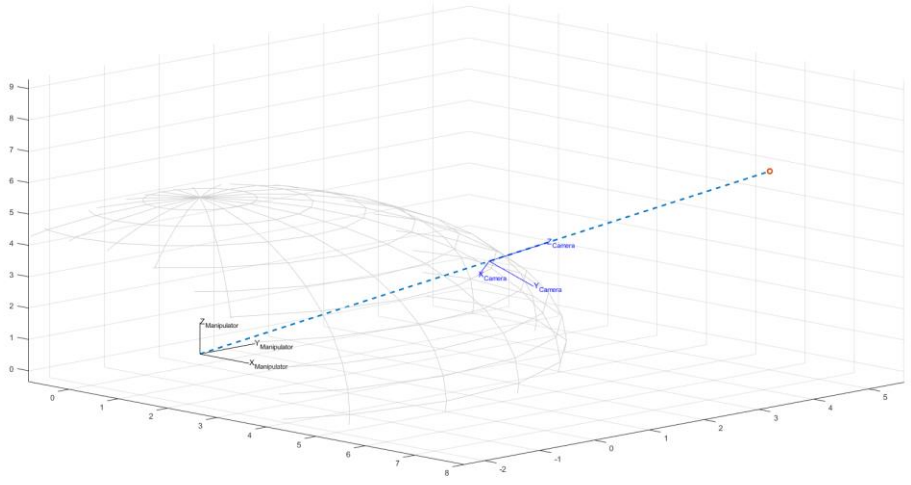
Řešení nedosažitelnosti nové polohy manipulátoru

Prvním krokem v řešení tohoto problému, je identifikovat nedosažitelnou pozici předtím, než takovou pozici pošleme manipulátoru. Pokud by se tak totiž stalo a manipulátor vyhodnotil obdrženou pozici jako invalidní, tak se nepohne, a navíc se zabrzdí. Kontrola nedosažitelnosti je prováděna zjednodušeně tak, že vezmeme novou polohu manipulátoru a zjistíme, jestli se nachází v kladné polokouli ($z>0$), která má střed v počátku souřadného systému manipulátoru a poloměr r.

Situace, kdy tedy máme nedosažitelnou novou pozici manipulátoru, je v rámci této práce řešena tak, že vypočteme nový směr a polohu kamery, aby byla dosažitelná a zanedbáme požadavek na udržení určité vzdálenosti. Toho docílíme následovně:

1. Určíme kulovou plochu uvnitř pracovního prostoru manipulátoru (celý jej objem se nachází uvnitř prostoru).
2. Na základě rovnice 2.4 vypočteme souřadnice středu objektu v souřadném systému manipulátoru, a zjistíme, zda se nachází mimo operační prostor.
3. Pokud je bod vně operačního prostoru, vypočítáme souřadnice průsečíku úsečky spojující střed naší kulové plochy a střed objektu s kulovou plochou.
4. Určíme natočení kolem os soustavy manipulátoru M tak, aby vrchní okraj snímku ležel v rovině, která je rovnoběžná s rovinou XY soustavy manipulátoru M a směr osy Z kamery směřoval na střed objektu. Když vezmeme v úvahu, jakým způsobem jsme si zvolili orientaci soustavy kamery, tak rovina ZY kamery by tedy měla být kolmá na rovinu XY manipulátoru.
5. Na základě poloh průsečíku a vypočtených úhlů sestavíme transformační matici H_{MCnew} (dle rovnice 2.21).
6. S využitím znalosti transformační matice H_{EC} vypočteme dle rovnice 2.11 transformační matici H_{MEnew}.

7. Z matice H_{MEnew} vyextrahujeme souřadnice nové polohy efektoru dle popisu na začátku této kapitoly (rovnice 2.12-2.15)



Obrázek 2.5 Ukázka nové polohy kamery na základě průsečíku přímky s koulí

Výpočet průsečíku s koulí

Uvažujme, že máme přímku danou dvěma body $P_1(x_1,y_1,z_1)$ a $P_2(x_2,y_2,z_2)$. Přímka P je pak popsána rovnic[21]

$$\mathbf{P} = \boldsymbol{P_1} + t(\boldsymbol{P_2} - \boldsymbol{P_1}) \,. \tag{2.16}$$

Koule má střed v bodě $P_c(x_c,y_c,z_c)$ a poloměr r a je popsána rovnicí[21]

$$(\boldsymbol{x} - \boldsymbol{x_c})^2 + (\boldsymbol{y} - \boldsymbol{y_c})^2 + (\boldsymbol{z} - \boldsymbol{z_c})^2 = \boldsymbol{r^2} \,. \tag{2.17}$$

Pokud dosadíme do rovnice koule za x, y a z příslušné prvky vektoru **P,** potom dostáváme kvadratickou rovnici[21]

$$\boldsymbol{at^2} + \boldsymbol{bt} + \boldsymbol{c} = \mathbf{0} \,. \tag{2.18}$$

Řešením této kvadratické rovnice dostaneme výsledné t, jehož dosazením do rovnice popisující naši přímku dostaneme průsečík/y této přímky s onou koulí. Počet řešení je odvislý od hodnoty diskriminantu kvadratické rovnice. Pokud je diskriminant menší než 0, tak přímka neprotíná kouli. Pokud je diskriminant roven 0, tak je přímka tečnou koule a má tedy s koulí pouze jeden společný bod (tečný). A pokud je diskriminant větší jak 0, tak přímka protíná kouli a výsledkem jsou dvě t, tedy i dva průsečíky koule a přímky. [21]

Určení úhlu natočení kamery

Abychom dosáhli požadovaného natočení kamery, jak je popsáno v bodě 4. postupu, musíme znát dva body. Prvním je střed koule $P_c(x_c,y_c,z_c)$, pro kterou počítáme průsečík. Druhým je jeden z dvojice bodů - buď průsečík nebo střed sledovaného objektu. Jelikož je v obou případech postup stejný, označíme tento bod P(x,y,z). Požadované úhly natočení pak dostaneme pomocí následujících rovnic

$$\alpha = -tan^{-1}\left(\frac{x-x_c}{y-y_c}\right), \tag{2.19}$$

$$\gamma = -cos^{-1}\left(\frac{z-z_c}{\sqrt{(x-x_c)^2+(y-y_c)^2+(z-z_c)^2}}\right), \tag{2.20}$$

kde α je natočení kolem osy Z (roll) a γ je natočení kolem osy X (yaw).

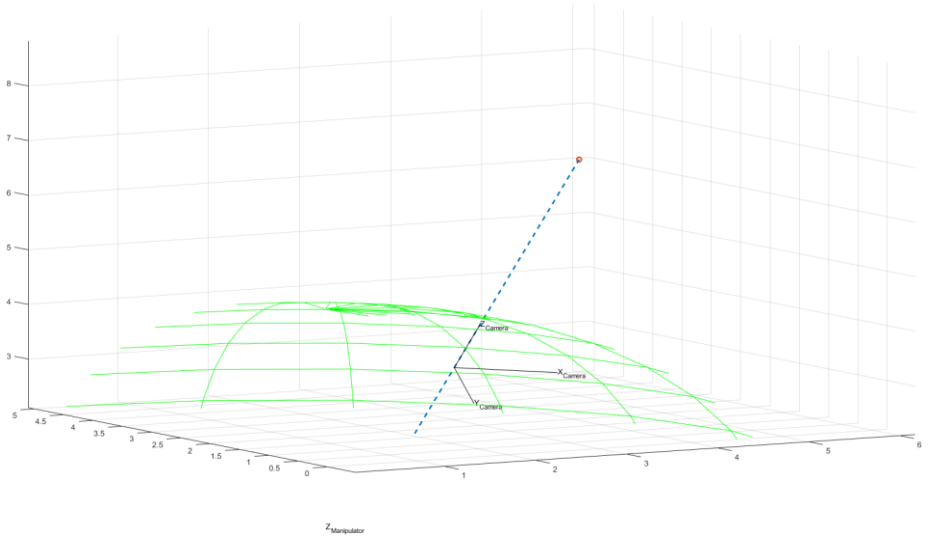
Výpočet H_{MEnew}

Zde na rozdíl od postupu výpočtu uvedeného v předchozí kapitole nepočítáme homogenní transformační matice H_{ME} a H_{MC}. V tomto případě přistoupíme rovnou k výpočtu homogenní transformační matice H_{MCnew} na základě následující rovnice

$$H_{MCnew} = Trans(x,y,z)Rot_z(\alpha)Rot_x(\gamma) = \begin{pmatrix}1&0&0&x\\0&1&0&y\\0&0&1&z\\0&0&0&1\end{pmatrix}\begin{pmatrix}c(\alpha)&-s(\alpha)&0&0\\s(\alpha)&c(\alpha)&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}\begin{pmatrix}1&0&0&0\\0&c(\gamma)&-s(\gamma)&0\\0&s(\gamma)&c(\gamma)&0\\0&0&0&1\end{pmatrix} = \begin{bmatrix}c(\alpha)&-s(\alpha)\cdot c(\gamma)&s(\alpha)\cdot s(\gamma)&x\\s(\alpha)&c(\alpha)\cdot c(\gamma)&-c(\alpha)\cdot s(\gamma)&y\\0&s(\gamma)&c(\gamma)&z\\0&0&0&1\end{bmatrix}, \tag{2.21}$$

kde c(x) = cos(x) a s(x) = sin(x), x, y a z jsou souřadnice průsečíku přímky a koule v souřadnicovém systému manipulátoru M, α je natočení kolem osy z (roll) a γ je natočení kolem osy x (yaw).

Dále se již postupuje stejně, jak je popsáno v předcházející kapitole. Tedy je z homogenních transformačních matic H_{MCnew} a H_{CE} určena transformační matice H_{MEnew}. A z této matice se poté získají souřadnice nové polohy efektoru, jak je uvedeno na začátku této kapitoly.



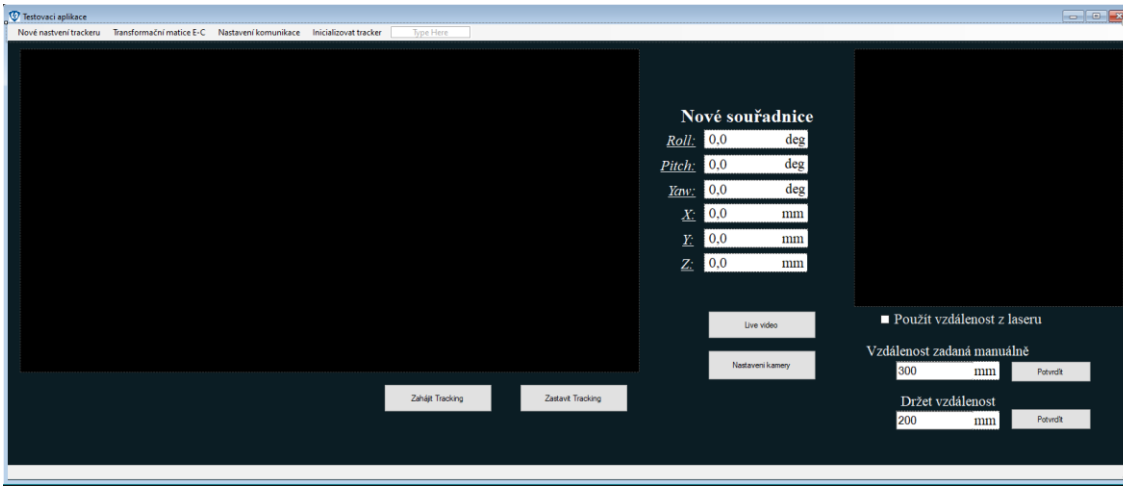
Obrázek 2.6 Ukázka orientace kamery v případě řečení pomocí průsečíku přímky a koule

2.6 Implementace na manipulátor

Jak plyne ze zadání této práce, její součástí je i implementace sledovacích algoritmů na manipulátor Epson C3. To je realizováno jednoduchou testovací aplikací v jazyce C# s rozhraním Windows Form.

Mimo aplikaci byla pro každou metodu vytvořena vlastní třída, jmenovitě jsou to FeatureMatchingTracker, MarkersTracker, OpenCvTracker. Všechny tři třídy mají společný interface TrackerInterface. Ten zajišťuje jednotné chování pro metody, které je volají.

2.6.1 Popis oken aplikace



Obrázek 2.7 Obrázek hlavního okna testovací aplikace

Hlavní okno aplikace

Okno, které se otevře jako první po spuštění aplikace. Obsahuje hlavní ovládací prvky aplikace.

Horní lišta:

- Nové nastavení trackeru – otvírá dialogové okno sloužící k nastavení nového trackeru
- Transformační matice E-C – otvírá dialogové okno sloužící k zadání homogenní transformační matice ze systému kamery do systéme efektoru
- Nastavení komunikace – otvírá dialogové okno sloužící k navázání pojení s manipulátorem
- Inicializovat tracker – spouští inicializační sekvenci trackeru

Spodní lišta:

- Zobrazuje, zda proběhla inicializace trackeru úspěšně, nebo je ji potřeba zopakovat

Nové souřadnice:

- Roll – zobrazuje roll úhel nové polohy efektoru
- Pitch – zobrazuje pitch úhel nové polohy efektoru
- Yaw – zobrazuje yaw úhel nové polohy efektoru
- X – zobrazuje x souřadnici nové polohy efektoru
- Y – zobrazuje y souřadnici nové polohy efektoru
- Z – zobrazuje z souřadnici nové polohy efektoru

Picture boxy:

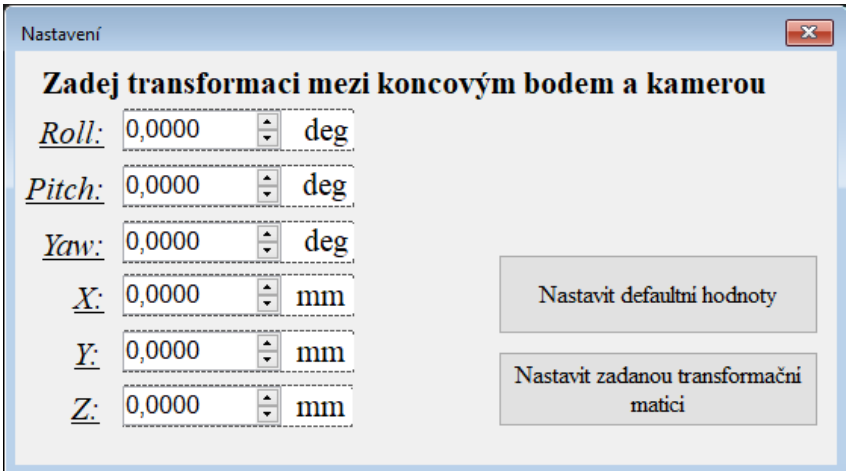
- Levý – zobrazuje výsledky detekce objektu v obraze
- Pravý – zobrazuje náhle živého přenosu z kamery

Vstupní boxy:

- Vzdálenost zadaná manuálně – box pro zadávání vzdálenosti kamery od předmětu manuálně
- Držet vzdálenost – box pro zadávání vzdálenosti jakou si má kamera držet od sledovaného objektu

Tlačítka:

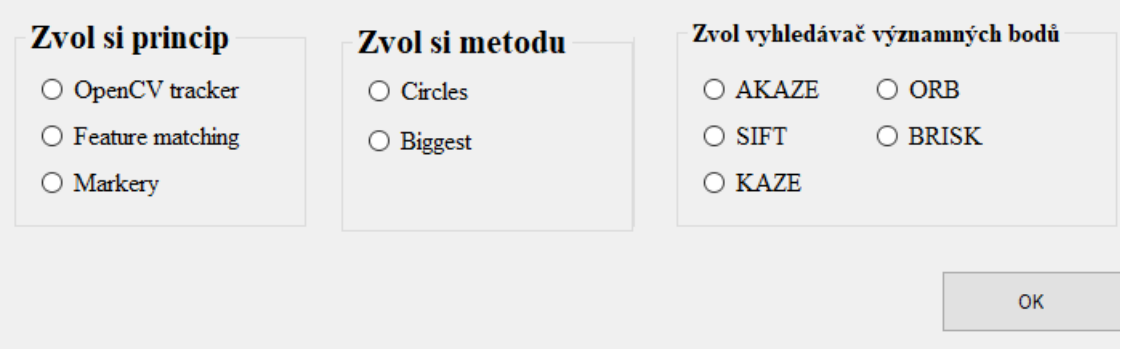
- Zahájit Tracking – spouští sledování objektu
- Zastavit Tracking – zastavuje sledování objektu
- Live video – pokud je kamera připojená, tak zahajuje živý přenos z kamery zobrazovaný v pravém Picture boxu.
- Nastavení kamery – vyvolává dialogové okno s nastavením parametrů kamery
- Použít vzdálenost z laseru – checkbox, který povoluje nebo zakazuje používat vzdálenost získanou z laserového skaneru
- Potvrdit – tlačítka, která potvrzují zadání/zápis hodnot, které jsou v boxech vedle nich



Obrázek 2.8 Obrázek okna nastavení transformace

Okno výběru typu trackeru

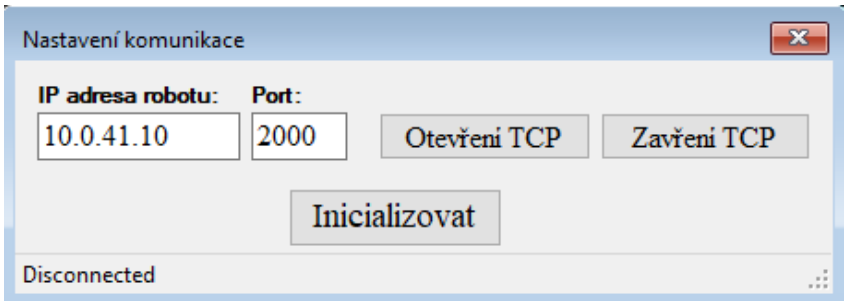
Dialogové okno, které uživateli umožní nastavit transformační matici z kamery do efektoru. Natočení lze zadat ručně do patřičných boxů, nebo lze načíst pomocí tlačítka *Nastavit defaultní hodnoty* defaultní hodnoty. Zadané hodnoty pak lze potvrdit tlačítkem *Nastavit zadanou transformační matici*.



Obrázek 2.9 Obrázek okna výběru typu trackeru

Okno výběru typu trackeru

Dialogové okno, které uživatele provede procesem výběru trackeru. V každém rámečku je možné vybrat jen jednu možnost. Jednotlivé rámečky mizí a objevují se na základě volby uživatele. Pokud uživatel označí všechny potřebné volby, objeví se tlačítko *OK*, kterým může uživatel výběr potvrdit a tím okno uzavřít



Obrázek 2.10 Obrázek dialogu pro nastavení komunikace

Okno výběru typu trackeru

Dialogové okno, které uživateli umožní navázat komunikaci s manipulátorem. Do boxů se zadává IP adresa manipulátoru a jeho komunikační port. Pomocí tlačítka *Otevřít TCP*, lze komunikaci navázat a tlačítkem *Zavřít TCP* navázanou komunikaci uzavřít. V levém spodním rohu se nachází indikátor stavu spojení. Pokud je navázáno spojení, lze tlačítkem *Inicializovat* inicializovat manipulátor.

2.6.2 Popis implementace kódu

V následující kapitole bude zmíněn stručný popis toho, jak je napsán kód pro Testovací aplikaci.

Výběr konfigurace nového trackeru

Na základě uživatelovy volby v dialogu pro výběr typu trackeru je inicializována proměnná tracker typu TrackerInterface na novou instanci třídy dle zvolených možností.

Nastavení transformační matice trackeru

Pokud uživatel v dialogovém okně, pro nastavení transformační matice, potvrdí nastavené souřadnice a natočení, dojde k zavolání metody `SetEfectorCameraMatrix` trackeru s parametry, které byly zadány uživatelem v rámci dialogového okna. Tímto je nastavena matice pro aktuální tracker.

```
/*****************************************************************
void SetEfectorCameraMatrix(float roll_E_C, float pitch_E_C, float yaw_E_C, float
x_E_C, float y_E_C, float z_E_C);
```

Nastavení spojení s manipulátorem a jeho povelování

Pro navázání, ukončení spojení, inicializaci manipulátoru a jeho povelování je použita třída `TcpC3Com`. Jedná se o třídu, jež byla získána z projektu RoScan3D. Autorem této C# komunikační knihovny a tohoto driveru je Ing. Martin Fireš [23]. Tato knihovna byla publikována v rámci jeho diplomové práce na Vysokém učení technickém v Brně, proto ji lze pro naše účely využít. Tato knihovna nám umožňuje komunikovat s programem v řídící jednotce manipulátoru, kde se nachází řídící program v jazyce SPEL+, přes ethernet. Pomocí TCP/IP protokolu tedy posíláme příkazy řídícímu programu v řídící jednotce a stejným způsobem od něj získáváme informace.

Inicializace trackeru

Stiskem příslušného tlačítka na horní liště hlavního okna aplikace dochází k navázání komunikace s kamerou a laserovým skenerem, pokud již s těmito zařízeními komunikace navázaná není. Pro operace s kamerou je použit driver společnosti The Imaging Source Europe GmbH a pro komunikaci s laserovým skenerem je využita komunikační knihovna, jejímž autorem je Ing. Adam Chromý, Ph.D. Komunikace s oběma zařízeními probíhá přes ethernet.

Pokud bylo spojení navázáno úspěšně, tak aktivujeme kameru a počkáme nějaký čas, než se její parametry ustálí. Pokud bychom nepočkali, první zachycený snímek by byl zcela přesvětlen. Po uplynutí stanovené doby získáme jeden aktuální snímek z kamery a ten použijeme jako parametr při volání inicializační metody pro aktuální tracker. Výsledek inicializační metody je poté indikován texem zobrazeným na spodní liště hlavního okna aplikace.

```
/*****************************************************************
bool initialize(Image<Bgr, byte> img);
```

Vyčítání obrázků z kamery

Vyčítání a zpracovávání obrazu z kamery je prováděno funkcí `colorCam_ImageAvailable`, která je obslužnou rutinou pro event třídy kamery, ten je vyvolán vždy, když je v bufferu

k dispozici nový obrázek z kamery. Pokud je zahájeno trackování a manipulátor není v pohybu, popřípadě se jedná o inicializaci, tak je obrázek z bufferu získán a zmenšen (1366x768) kvůli snížení výpočetní náročnosti. Pro zmenšení obrázku je využita metoda interpolace nejbližším sousedem. Tato metoda byla zvolena, protože zachovává hrany v obraze, a to je často klíčový prvek pro naše vyhledávací algoritmy. Ostatní metody interpolací totiž používají metody průměrování hodnot z nějakého okolí, a tím filtrují obraz. Ten tak přichází o své hrany. Zmenšený obrázek je pak uložen do lokální proměnné, aby mohl být dále využit pro sledování.
V případě, kdy je sledování aktivní a nastane event kamery `ImageAvailable`, tak je načtený obrázek z bufferu zmenšen, uložen a následně je vyvolán event `imageLoadedForTracking`.

Výpočet nové pozice manipulátoru na základě snímku z kamery
Výpočet nastává, pokud je vyvolán event `imageLoadedForTracking`. Tento event je obsluhován funkcí `moveWithManipulator`. V rámci této funkce dojde nejprve k vyžádání aktuální pozice ramene z manipulátoru. Dále pokud je zvolena možnost využívání informací o vzdálenosti objektu z laserového skeneru, tak jsou z něj vyčtena data. Pokud se data ze skeneru podaří vyčíst, jako vzdálenost předmětu od kamery je označena souřadnice z, takového bodu aktuálního profilu skeneru, jehož souřadnice x je první menší nebo rovna -7 mm. Pokud žádný bod v aktuálním profilu neodpovídá zmíněné podmínce, vezme se bod co leží nejvíce vpravo, tedy bod s indexem 0. Důvodem pro tuto volbu je, že laserový paprsek protíná rovinu YZ kamery, proto se tento bod nachází nejblíže středu. V ostatních případech je nejblíže středu kamery krajní bod. Poté čekáme, dokud neobdržíme odpověď od manipulátoru s aktuálními souřadnicemi koncového bodu. Až je obdržíme zavoláme metodu `GetNewEfectorCoordiantes` trackeru, která na základě předaných souřadnic efektoru, aktuálního snímku, aktuální vzdálenosti, vzdálenosti, kterou si má kamera udržovat a několika dalších parametrů vypočte nové souřadnice efektoru jako výstupní argument. Její návratová hodnota značí, zda se výpočet zdařil, či nikoliv.

Vyslání manipulátoru na novou pozici
Pokud proběhne výpočet nových souřadnic úspěšně, znamená to, že nové souřadnice jsou nenulové. V tu chvíli dojde k vykreslení geometrických útvarů, značících místo výskytu objektu, do scény, na jejímž základě byly nové souřadnice počítány. Tato scéna je pak zobrazena v levém PictureBoxu ve hlavním okně aplikace. Poté se zkontroluje, zda je navázáno spojení s manipulátorem. Pokud tomu tak je, pošle se požadavek pomocí metody `SendGo` do manipulátoru s novými souřadnicemi. Poté se čeká, dokud manipulátor nedosáhne požadované polohy. V tu chvíli začne sledovací cyklus od začátku.

2.6.3 Popis ovládání aplikace

Po startu aplikace je potřeba nakonfigurovat nový tracker. Stiskneme tedy tlačítko na horní liště *Nové nastavení trackeru* a objeví se nám průvodce nastavením trackeru. Pokud jednou zahájíme nastavování, nemůžeme z něj odejít jinak, než ho dokončit. V okně nejprve zvolíme požadovaný princip sledování. Dle zvoleného principu se objeví další možnosti volby, které mohou mít ještě více podvoleb. V každém okénku lze zvolit pouze jednu z možností. Pokud jsou navoleny všechny potřebné možnosti, zobrazí se tlačítko OK pro potvrzení výběru a po jeho stisknutí je výběr hotov.

Dále je třeba nastavit trackeru homogenní transformační matici z kamery do efektoru. Toho docílíme stiskem tlačítka *Transformační matice E-C* na horní liště. Po kliknutí se zobrazí okno, ve kterém je možné zadat jednotlivé údaje o transformaci do příslušných polích ve formátu roll, pitch, yaw, x, y, z. Abychom nemuseli v rámci testování zadávat parametry ručně, lze načíst defaultní hodnoty do polí pro námi testovanou soustavu kliknutím na tlačítko *Nastavit defaultní hodnoty*. Po zadání hodnot nebo jejich načtení hodnoty potvrdíme stiskem tlačítka *Nastavit zadanou transformační matici* a tím je matice pro tracker nastavena.

Máme-li nastavenou transformační matici, je na řadě navázat spojení s manipulátorem. Nejprve tedy stiskneme na tlačítko *Nastavení komunikace* na horní liště, aby se nám otevřelo okno pro nastavení komunikace. V tomto okně vyplníme IP adresu robotu a jeho komunikační port, s kterým chceme komunikovat. Po nastavení těchto hodnot stiskneme tlačítko *Otevřít TCP* pro navázání TCP spojení s robotem. Vedle tohoto tlačítka se nachází tlačítko, které slouží pro případné zavření aktivního TCP spojení. Stav TCP spojení je zobrazen v levém spodním rohu. Pokud se nám podařilo připojit k manipulátoru, můžeme ho inicializovat kliknutím na tlačítko *Inicializovat* (pokud ho chceme používat, tak bychom inicializovat měli). Po úspěšné inicializaci (zvuk odbrzdění manipulátoru) můžeme okno s nastavením komunikace zavřít.

Posledním krokem, před tím než je možné spustit sledování, je inicializace trackeru. Tu provedeme opět stiskem tlačítka *Inicializovat tracker* na horní liště. Po kliknutí je nutné chvíli vyčkat, až po několika sekundách se objeví okno se snímkem z kamery. Po nás, jakožto po uživateli, se pak vyžaduje abychom vybrali objekt, který chceme sledovat. Pokud jsme na začátku vybrali OpenCV nebo Feature matching tracker, tak kurzorem myši kliknutím a popotažením vybereme oblast, kde se na snímku nachází náš objekt zájmu. Pokud jsme vybrali sledování markerů, vybereme stejným způsobem vnitřek markeru. Výběr potvrdíme stiskem klávesy „space“ nebo „enter“. Nejsme-li spokojeni s kvalitou obrazu a nastavením kamery, můžeme okno zavřít stiskem klávesy „c“. Poté můžeme nastavení kamery změnit tak, že klikneme na tlačítko *Nastavení kamery* a otevře se nám okno s parametry kamery, které chceme měnit. Jestliže chceme vidět, jak se změny, které provádíme s parametry, projevují v obraze, můžeme stiskem tlačítka *Live video* spustit v pravém horním rohu hlavního okna živý přenos z kamery. Poté, co jsme již spokojeni s nastavením kamery, dialogové okno potvrdíme tlačítkem OK a zavřeme.

Dále zopakujeme proceduru inicializace trackeru. Informace o tom, zda se inicializace zdařila či nikoli, se objeví v levém spodním rohu na liště hlavního okna.

V tuto chvíli je již možné spustit sledování tlačítkem *Zahájit Tracking*. Před spuštěním sledování je však doporučeno změřit a nastavit vzdálenost mezi kamerou a objektem ručně do pole *Vzdálenost zadaná manuálně* a potvrdit ji tlačítkem *Potvrdit*. Nebo zaškrtnout pole *Použít vzdálenosti z laseru*, to je však doporučeno převážně pouze pro sledování nad nějakou deskou, stolem, kdy jsou rovnoběžné s rovinou XY manipulátoru a kamera se na ni dívá kolmo. Laser totiž nemá ošetřené všechny chybové stavy pro měření mimo rozsah. Pokud zvolíme zadávání vzdálenosti manuálně, je nezbytně nutné zadat do pole *Držet vzdálenost* stejnou vzdálenost, jakou jsme zadali do pole *Vzdálenost zadaná manuálně* nebo nulovou vzdálenost a potvrdit sousedním tlačítkem *Potvrdit*. Pokud bychom tak neučinili, manipulátor by se neustále snažil vzdálenost zkracovat nebo se vzdalovat podle rozdílu v těchto zadaných vzdálenostech. Hrozil by tak náraz manipulátoru do jiného objektu. V případě, že vybereme a zaškrtneme možnost získávání dat z laserového skeneru, můžeme zadat do pole *Držet vzdálenost* vzdálenost v rozmezí měřícího rozsahu skeneru (190-290 mm) a tu poté potvrdit.

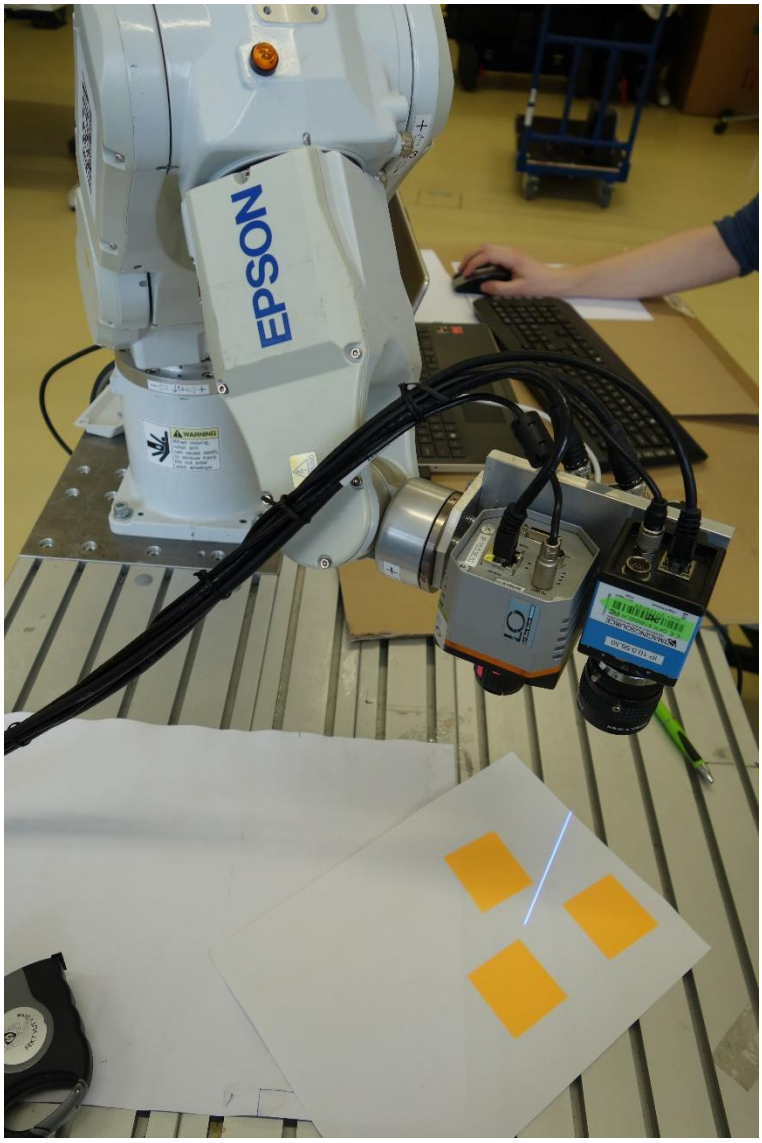
Nyní tedy můžeme zahájit sledování stiskem tlačítka *Zahájit Tracking*. Během spuštěného sledování je vhodné mít po ruce stop tlačítko od manipulátoru, jelikož program nemá řešeny a ošetřeny všechny případy kolizí.

V průběhu sledování je možné v hlavním okně vidět v levé horní části průběh sledování s detekovaným umístěním předmětu a napravo od toho vypočtené nové souřadnice efektoru manipulátoru.

Zastavit sledování lze stiskem tlačítka *Zastavit Tracking*. Zastavit sledování je nutné i v případě, že chceme nastavit jiný trackovací algoritmus. V tom případě sledování zastavíme a musíme celý postup zopakovat od začátku.

2.6.4 Výsledky testování

Bylo provedeno několik testování funkčnosti a výkonnosti aplikovaných algoritmů přímo při nasazení na manipulátoru. Algoritmy pro sledování fungovaly a plnily svou funkci, tedy polohovali manipulátor tak, aby se střed sledovaného objektu držel na středu kamery. Výsledky těchto testů budou rozebrány ve zbytku této kapitoly. Videa zachycující průběh některých testování je možno nalézt v příloze C.



Obrázek 2.11 Fotografie z průběhu testování algoritmů

Polohování robota

Polohování robota doprovázely jisté neduhy. Vzhledem k tomu, že polohování do nové polohy zajišťujeme zasíláním nových souřadnic koncového bodu manipulátoru a způsob jakým se do tohoto bodu manipulátor dostane je ponechán na vnitřní logice jeho řídící jednotky, dochází v některých místech pracovního prostoru k častým přetáčením kloubů, i když by to nebylo zcela nezbytné.

Dalším problémem polohování se stala bezpečnost, jelikož ta je v algoritmech zakotvena jen velmi zjednodušeně. Proto není například řešena případná kolize držáku s kamerami a skenerem, s ostatními částmi ramene robota a některé další věci.

Tyto problémy by mohly být minimálně částečně řešeny tak, že bychom nepolohovali na základě polohy efektoru, ale prováděli bychom inverzní kinematickou úlohu a řídili bychom manipulátor pomocí úhlového natočení jednotlivých kloubů. To by nám poskytovalo větší kontrolu nad manipulátorem a usnadnilo implementaci některých bezpečnostních pravidel.

Rychlost

Pokud chceme posuzovat rychlost, musíme vzít také v potaz omezení manipulátoru, kdy musíme vždy počkat, než dokončí manipulátor pohyb, abychom mohli manipulátoru zaslat nové souřadnice. Proto při posuzování rychlosti polohování manipulátoru ovládaného vstupem z kamery je třeba vzít v úvahu, že jsme nikdy netestovali, vzhledem k bezpečnosti, algoritmy při full-power módu manipulátoru, takže výsledky nemusí být zcela autentické. Stačilo to však, abychom si o výkonech algoritmů udělali jistou představu, o kterou se zde podělíme.

Při přímém nasazení byly všechny algoritmy poměrně pomalé. Nejpomalejší byly OpenCv trackovací algoritmy. Jejich rychlost se odvíjela především od toho, jak velkou část snímku zabíral sledovaný objekt. Čím větší část snímku objekt zabíral, tím byly algoritmy pomalejší. Algoritmus CSRT byl v některých případech tak pomalý, že mu cyklus výpočtu mohl trvat i 1-3 sekundy. KCF algoritmus byl rychlejší, ale ani jeho rychlosti z daleka nedosahovaly hodnot, které bychom potřebovali pro real-time sledování.

V případě Feauture matching metod jsme dosahovali obdobných rychlostí jako u KCF trackeru nebo o něco rychlejších výsledků. Rychlejší byly především metody využívající jeden z trojice algoritmů na vyhledávání a popis významných bodů SIFT, ORB, KAZE.

Sledování markerů vzhledem k tomu, že využívá nejjednodušší algoritmus, dosahovalo nejvyšších rychlostí, kdy se nevíce blížilo plynulému sledování. Ani to by pravděpodobně nestačilo pro plynulé sledování při full-power módu manipulátoru.

Spolehlivost detekce

Všechny algoritmy ve většině případů dokázaly zpočátku sledovat objekt bez větších problémů, po čase se ovšem mohlo stát, že byl sledovaný objekt ztracen. To nastávalo z několika různých příčin. Všechny z použitých algoritmů se také zdárně vyvarovávaly falešně pozitivních detekcí, až na jednotlivé případy. Častějším problémem se stala spíše neschopnost detekovat hledaný objekt v obraze.

Ukázalo se, že algoritmy pro sledování markerů a algoritmy na bázi feature matchingu značně závisí na světelných podmínkách a na správném nastavení kamery. Používaná kamera je totiž sama o sobě poměrně citlivá na světelné podmínky, kdy se změnou těchto podmínek se značně mění i parametry snímané scény. To má za důsledek ztráty sledovaného předmětu u zmíněných dvou algoritmů. Při správném nastavení kamery a stálých světelných podmínkách jsou však zmíněné algoritmy spolehlivé a přesné.

Spolehlivost OpenCV trackovacích algoritmů závisí především na počátečním nastavením kamery. Pokud je kamera dobře nastavena a scéna není tmavá ani přesvětlená a je dostatečně kontrastní, tak jsou OpenCV algoritmy velice přesné a spolehlivě detekují objekt. V jejich případě nastává problém při velkých pohybech mezi jednotlivými snímky, kdy ztrácí sledovaný objekt. Důvodem je pravděpodobně fakt, že jelikož jsou to sledovací algoritmy v pravém slova smyslu, tak si vytvářejí pohybový model a pokud dojde k velkému pohybu mezi snímky, tak nová poloha objektu nesouhlasí s pohybovým modelem a objekt je tedy ztracen. Objekt může být poté znovu nalezen, pokud se vrátí na místo, kde ho detektor naposledy detekoval. OpenCV algoritmy se s časem stráveným úspěšným sledováním objektu zlepšují, jak již vyplývá z jejich principů, a stávají se tak robustnějšími, méně náchylnými ke změnám ve snímané scéně. Mají také velmi statický ustálený stav detekce (manipulátor se moc nepohybuje a je zaměřen na objekt).

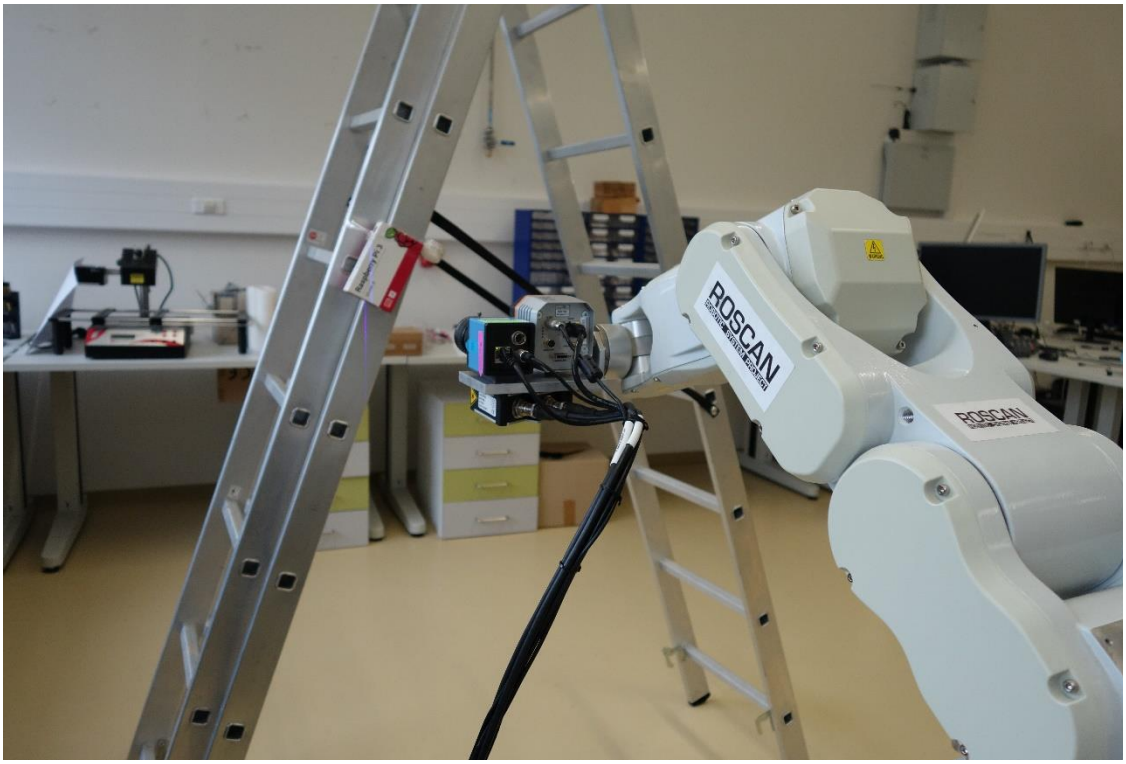
Konfigurace algoritmů s nejlepšími výsledky

Nejlepších souhrnných výsledků ve vícero kategoriích dosahovaly algoritmy v následujících kombinacích:

- OpenCV tracker – KCF (spolehlivý, poměrně rychlý, po delší době sledování se stává robustním)
- OpenCV tracker – KCF (velmi spolehlivý, velice přesný, značně pomalý, po delší době sledování se stává robustním, ztráty při velkých pohybech)
- Feature matching – FLANN – SIFT (spolehlivá detekce, poměrně odolný proti falešně pozitivním nálezům)
- Feature matching – Brute Force – ORB (rychlý, poměrně spolehlivý)
- Markery – Circles (vysoká rychlost, velká odolnost proti falešně pozitivním nálezům)
- Markery – Biggest (nejrychlejší, slušná spolehlivost, můžou se vyskytnout falešně pozitivní nálezy)

Ověření funkčnosti jednotlivých funkcí

Úspěšně jsme ověřili jak funkčnost detekce objektů pomocí algoritmů, tak schopnost na základě těchto informaci správně a úspěšně přesunout manipulátor do požadované polohy. V případě markerového sledování jsme byli schopni také udržovat konstantní natočení markerů. Byla ověřena i funkčnost udržování konstantní vzdálenosti od sledovaného objektu za pomoci laserového skeneru. Tato funkcionalita nefunguje za všech podmínek, především mimo měřící rozsah skeneru a pokud dojde k natočení nebo posunu objektu tak, že na něj nedopadá laserový paprsek. Tyto chybové stavy invalidních stavů/naměřených hodnot laserového skeneru nejsou v rámci testovací aplikace ošetřeny. Úspěšně jsme rovněž ověřili chování pro nedosažitelné stavy, kdy je poloha kamery počítána na základě průsečíku přímky s koulí a dívá se směrem na střed sledovaného objektu. Jednotlivá videa dokazující tato tvrzení lze nalézt v příloze C.



Obrázek 2.12 Ukázka testování sledování objektů mimo dosah

Využitelnost v medicíně a fotografování

Otázka využitelnosti algoritmu v této práci je poměrně jednoduchá. V současné podobě by algoritmus neměl využití ani v medicíně ani ve fotografování. Proto aby mohl být využit v medicíně by bylo potřeba doladit bezpečnostní podmínky a funkce a zvýšit lehce výkon algoritmů. V tomto případě zde není tak velký rozdíl v aktuálním výkonu a výkonu, který by postačoval pro medicínské účely.

Co se týče využití ve fotografování, tak pokud by šlo pouze o zacílení a pořízení fotografie, zde by mohl být algoritmus využit, žádné. Ovšem původní myšlenka na začátku této práce byla využití pro natáčení objektů v pohybu. V této oblasti se značně stávajícím algoritmům nedostává potřebného výkonu, aby mohly zajistit plynulé natáčení objektů v pohybu. Zde se ani nenaskytuje v současné době možnost nějaké jednoduché nebo rychlé nápravy či úpravy, která by tuto činnost umožnila.

3. ZÁVĚR

Tato diplomová práce byla velmi prakticky zaměřená a pokrývala široké spektrum témat od počítačového vidění po robotické manipulátory. Proto bylo nezbytné seznámit se s přístroji jako byl robotický manipulátor a kamera, na které jsem nakonec implementoval své algoritmy.

Jako softwarový nástroj pro vývoj algoritmů počítačového vidění jsem zvolil široce využívanou knihovnu OpenCV, přesněji tedy její wrapper knihovnu EmguCV. S využitím funkcí počítačového vidění, které tato knihovna poskytuje, jsem zvolil a implementoval tři různé přístupy k problematice sledování objektů v obraze.

Prvním jsou trackovací algoritmy, které jsou již implementovány v knihovně. Druhým je detekování sledovaného objektu v každém jednotlivém snímku pomocí klasických detektorů a deskriptorů významných bodů a následného feature matchingu, hledání shody mezi významnými body. Třetím přístupem bylo sledování markerů o předem známé a definované podobě.

Dalším bodem důležitým pro funkčnost úlohy sledování objektu pomocí robotického manipulátoru je výpočet nové polohy manipulátoru. Proto jsem vytvořil postup pro výpočet nové polohy koncového dobu manipulátoru. Ten byl založen na pravidlech pro výpočet homogenních transformací mezi jednotlivými částmi řetězce manipulátoru.

Sledovací algoritmy a postup výpočtu nové polohy jsem poté přetvořil do tříd, tak aby mohly být implementovány v testovací aplikaci implementované přímo na reálném manipulátoru.

Výsledky testování však nedostály počátečním očekáváním. Původním záměrem bylo vytvořit algoritmy použitelné v medicínském skenování a fotografování. Což se mi nepodařilo zrealizovat v praxi. Samotný princip sledování pomocí robotického manipulátoru a kamery jsem zvládl uskutečnit, přesnost a spolehlivost použitých algoritmů byla ve většině případů uspokojivá. Avšak algoritmy značně zaostávaly za požadovanou rychlostí, která by zaručovala hladký a plynulý průběh sledování. Tato skutečnost v řadě případů zpětně negativně ovlivňovala spolehlivost a přesnost detekce v případech rychlých pohybů nebo velkých vzdáleností od kamery. Problémem se také stala velká natočení mimo rovinu XY obrazu. Bylo tedy sice splněno zadání této práce, ale nepřiblížil jsme se k cíli, který jsme si na začátku vytyčili a chtěli ho dosáhnout.

Ze zkušeností, které jsem v rámci této diplomové práci nabyl, vyplývá fakt, že algoritmy počítačového vidění pro univerzální použití, jaké jsem vytvářel v rámci této práce, nemají zpravidla dostatečně velkou přesnost a spolehlivost. Proto je vhodnější pro aplikace, kde na těchto parametrech záleží, úzce specifikovat problém a vyvíjet specializované algoritmy pro specifické použití. Ty pak mají sice jen velmi úzké využití, ale dosahují výsledků, které s univerzálními algoritmy nedosáhneme.

LITERATURA

[1] *EPSON ProSix C3 series MANIPULATOR MANUAL* [online]. Rev.9. EPSON, 2015 [cit. 2021-01-03]. Dostupné z: https://files.support.epson.com/far/docs/epson_c3_robot_manual(r9).pdf

[2] *Epson RC180 Controller Manual* [online]. Rev.15. EPSON, 2013 [cit. 2021-01-03]. Dostupné z: https://files.support.epson.com/far/docs/epson_rc180_controller_manual(r15).pdf

[3] *ISO 8373:2012: Robots and robotic devices — Vocabulary*. International Organization for Standardization, 2012.

[4] ŠOLC, František a Luděk ŽALUD. *ROBOTIKA* [online]. Brno: VUT, 2006 [cit. 2021-01-03]. Dostupné z: https://www.uamt.feec.vutbr.cz/~robotika/prednasky/Robotika.pdf

[5] SKAŘUPA, Jiří. *ROBOTY A MANIPULÁTORY* [online]. Ostrava: VŠB - Technická univerzita Ostrava, 2012 [cit. 2021-01-03]. Dostupné z: http://www.person.vsb.cz/archivcd/FS/RaMa/Roboty%20a%20manipulatory.pdf

[6] JELÍNEK, Aleš a Adam CHROMÝ. *Vybrané partie z robotiky* [online]. Brno: VUT v Brně, 2015 [cit. 2021-01-03]. Dostupné z: https://www.uamt.feec.vutbr.cz/~robotika/Vybrane_partie_z_robotiky.pdf

[7] VOJÁČEK, Antonín. Princip laserových snímačů vzdálenosti s triangulačním principem měření. *Automatizace.hw.cz* [online]. 2015, 13. Červenec 2015 [cit. 2021-01-04]. Dostupné z: https://automatizace.hw.cz/mereni-a-regulace/princip-funkce-laserovych-snimacu-vzdalenosti-s-triangulacnim-principem-mereni.html

[8] Laserové triangulační snímače. *AUTOMA* [online]. Ústí nad Labem: AUTOMA, 2013, 12/2013, (12), 25. str [cit. 2021-01-04]. Dostupné z: https://automa.cz/Aton/FileRepository/pdf_articles/10893.pdf

[9] 1D Laser Displacement Sensors. *Measurement Library* [online]. KEYENCE, © 2021 [cit. 2021-01-04]. Dostupné z: https://www.keyence.eu/ss/products/measure/measurement_library/type/laser_1d/

[10] *Data sheet scanCONTROL 29x0* [online]. Ortenburg / Germany: MICRO-EPSILON Headquarters [cit. 2021-5-17]. Dostupné z: https://www.micro-epsilon.cz/download/products/cat-scancontrol/dax--scanCONTROL-29x0--en.html#page=2&zoom=Fit

[11] MALLICK, Satya. Object Tracking using OpenCV (C++/Python). *Learn OpenCV* [online]. © 2020, 13.2.201 [cit. 2021-01-04]. Dostupné z: https://www.learnopencv.com/object-tracking-using-opencv-cpp-python/

[12] Homogeneous coordinates. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2021 [cit. 2021-01-04]. Dostupné z: https://en.wikipedia.org/wiki/Homogeneous_coordinates

[13] Euler angles. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2021, 25.12.2020 [cit. 2021-01-04]. Dostupné z: https://en.wikipedia.org/wiki/Euler_angles

[14] IVS ImagineSource Catalog. IVS imaging, 2012.
[15] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. Visual tracking with online multiple instance learning. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 983–990. IEEE, 2009.
[16] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In *proceedings of the European Conference on Computer Vision*, 2012.
[17] Alan Lukezic, Tom'as Voj'ir, Luka Cehovin Zajc, Jir'i Matas, and Matej Kristan. Discriminative correlation filter tracker with channel and spatial reliability. *International Journal of Computer Vision*, 2018.
[18] David S. Bolme, J. Ross Beveridge, Bruce A. Draper, and Man Lui Yui. Visual object tracking using adaptive correlation filters. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.
[19] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(7):1409–1422, 2012.
[20] PAL, Shukant. *Understanding 3D matrix transforms* [online]. 2019 [cit. 2021-5-17]. Dostupné z: https://medium.com/swlh/understanding-3d-matrix-transforms-with-pixijs-c76da3f8bd8
[21] BOURKE, Paul. *Circles and spheres* [online]. April 1992 [cit. 2021-5-17]. Dostupné z: http://paulbourke.net/geometry/circlesphere/index.html#linesphere
[22] HENDERSON, D.B. *Euler Angles, Quaternions and Transformation Matrices* [online]. Huston, Texas: NASA, 1977 [cit. 2021-5-17]. Dostupné z: https://ntrs.nasa.gov/api/citations/19770024290/downloads/19770024290.pdf?attachment=true
[23] FIREŠ, M. Demonstrační úloha pro robotický manipulátor EPSON. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2012. 73 s. Vedoucí diplomové práce doc. Ing. Luděk Žalud, Ph.D..

SEZNAM SYMBOLŮ A ZKRATEK

Zkratky:

ASCII	American Standard Code for Information Interchange
BL	Blue (modrý laser)
BSD	Berkeley Software Distribution
BRG	Modré (**B**lue), červené (**R**ed), zelené (**G**reen) – barevný prostor
CCD	Charge-coupled device
CMOS	*Complementary Metal–Oxide–Semiconductor*
CPU	Centrální procesorová jednotka
CSRT	Discriminative Correlation Filter with Channel and Spatial Reliability
CSR-DCF	Discriminative Correlation Filter with Channel and Spatial Reliability
CUDA	Compute Unified Device Architecture – hardwarová a softwarová architektura
DOF	Depth of field – rozsah vzdálenosti
FEKT	Fakulta elektrotechniky a komunikačních technologií
FLANN	Fast Library for Approximate Nearest Neighbors
FPS	F*rames per second*
GPU	Graphics processing unit – grafický procesor
GPGPU	General-purpose computing on graphics processing units
HSB	Hue, Saturation, Brightness – barevný prostor
HSV	Hue, Saturation, Value – barevný model
Hz	Herz, jednotka kmitočtu
ID	Osobní číslo
iOS	Mobilní operační systém
IP	Internet Protocol
ISO	International Organization for Standardization-Mezinárodní organizace pro normalizaci
KCF	**Kernelized Correlation Filters** -kernelizované korelační filtry
Kg	kilogram
LED	Light-Emitting Diode, česky elektroluminiscenční dioda, též světelná dioda
MIL	Multiple Instance Learning
m^2	metr tvereční

max	maximální/maximálně
mm	milimetr
mW	miliWatt
N.m	Jednotka momentu síly
ORB	Oriented FAST and Rotated BRIEF
PC	Personal Computer – osobní počítač
PLC	Programmable Logic Controller-programovatelný logický automat
RANSAC	Random sample consensus- koncensus náhodných vzorků
RGB	R-red (červená), G-green (zelená) a B-blue (modrá)
RTU	Remote Terminal Unit
S	Sekunda
SIFT	Scale-invariant feature transform
TCP	Transmission Control Protocol - protokol transportní vrstvy v sadě protokolů
TCT IP	Transmission Control Protocol/Internet Protocol – „primární přenosový protokol/protokol síťové vrstvy“
USB	Universal Serial Bus - univerzální sériová sběrnice
W	***Watt***, ***W*** , hlavní jednotka výkonu
UDP	User Datagram Protocol, Uživatelský protokol pro přenos datagramů v sítích
VUT	Vysoké učení technické v Brně

SEZNAM PŘÍLOH

Příloha A - Tracker Interface

A.1 Členy rozhlaní

```
double cameraAngleX { get; set; }
```

Zorný úhel kamery horizontální.

```
double cameraAngleY { get; set; }
```

Zorný úhel kamery vertikální.

```
double objectDistToKeep { get; set; }
```

Vzdálenost, kterou má kamera držet od sledovaného předmětu.

```
double sphereRadius { get; set; }
```

Poloměr kulového prostoru, který definuje pracovní prostor manipulátoru.

A.2 Metody rozhraní

```
bool initialize(Image<Bgr, byte> img);
```

Metoda provádějící inicializaci trackeru. Při inicializaci je vyžadováno po uživateli vybrat oblast výskytu sledovaného objektu z obrázku **img**, který je předán jako parametr. Úspěch či neúspěch inicializace je vracen jako návratová hodnota (true = úspěch, false = neúspěch).

```
void SetEfectorCameraMatrix(float roll_E_C,
      float pitch_E_C, float yaw_E_C, float x_E_C, float y_E_C,
      float z_E_C);
```

Metoda provádějící nastavení homogenní transformační matice ze soustavy kamery do soustavy efektoru. Zkonstruovaná matice má pořadí natočení roll, pitch, yaw, kdy roll je parametr **roll_E_C**, pitch je parametr **pitch_E_C**, yaw je parametr **yaw_E_C**. Translace matice x, y, z je dána parametry metody, kdy x je **x_E_C**, y je **y_E_C**, z je **z_E_C.**

```
bool GetNewEfectorCoordiantes(Image<Bgr, byte> img,
      double objectDist, float roll_B_E, float pitch_B_E,
      float yaw_B_E, float x_B_E, float y_B_E, float z_B_E,
      out RobCoord.XYZ newSysCoor, double cameraAngleX =
      60.7974554*Math.PI/180, double cameraAngleY =
      47.498989*Math.PI/180, double objectDistToKeep = 0,
      double sphereRadius = 800);

bool GetNewEfectorCoordiantes(Image<Bgr, byte> img,
      double objectDist, float roll_B_E, float pitch_B_E, float
      yaw_B_E, float x_B_E, float y_B_E, float z_B_E, out
      out RobCoord.XYZ newSysCoor);
```

Tyto dvě metody výše dělají plní stejnou funkci, pouze jedna přetěžuje druhou, v případě, že chce programátor zadat některé nepovinné argumenty. Tyto metody slouží k výpočtu nových souřadnic na základě obrazové informace poskytované parametrem **`img`**. Metodě je dále předáván parametr vzdálenosti kamery od objektu **`objectDist`**, parametry akruální pozice koncového bodu manipulátoru, kdy roll je **`roll_B_E`**, pitch je **`pitch_B_E`**, yaw je **`yaw_B_E`**, x je **`x_B_E`**, y je **`y_B_E`**, z je **`z_B_E`**. Nové souřadnice jsou vraceny pomocí návratového parametru **`newSysCoor.`** Dalšímy parametry jsou horizontální úhel záběru kamery **`cameraAngleX`** , vertikální úhel záběru kamery **`cameraAngleY`**, vzdálenost, kterou má držet kamera od objektu **`objectDistToKeep`** a poloměr kulovité oblasti, která udává pracovní prostor manipulátoru. Pokud je parametr nastaven na 0, tak kamera drží aktuální vzdálenost k objektu, tzn kompenzace vzdáleností je zanedbána. Úspěch či neúspěch výpočtu souřadnic je vracen jako návratová hodnota (true = úspěch, false = neúspěch).

Příloha B - Zdrojový kód programu je uložen na přiloženém DVD

Příloha C - Videa a obrázky dokumentující testování algoritmů na manipulátoru jsou uložena na přiloženém DVD